SKROLLI

2013.3
SKROLLI.FI
Tietokonekulttuurin erikoislehti

OHM & Assembly
Hakkerikekkereiltä pelihelvettiin

Oculus Rift
Lumetodellisuuden airut

Tietokone tietäjänä
Ymmärtääkö tekoäly?

Ohjelmointisarjat jatkuvat
QML ja OpenGL

Sisällysluettelo

Mikko Heinonen
asiantuntijatoimittaja

Yhdessä, ilosta

Skrollia tehdään, koska se on sen arvoista. Teille ja meille.

Vajaa vuosi sitten lähdin mukaan tekemään uutta tietokonekulttuurin erikoislehteä – siitäkin huolimatta, että elämän muiden kiireiden vuoksi ei olisi ehkä pitänyt haalia enää yhtään lisävastuita. Koska kasvoin lukien 80-luvun harrastuslehtiä ja selaan niitä ajoittain edelleen, oli ajatus jonkin samanlaisen synnyttämisestä aivan liian houkutteleva.

En tiennyt tuolloin, kuinka monta numeroa – jos sitä yhtäkään – tätä lehteä ilmestyisi. Sitten tutustuin muihin toimituksen jäseniin, joista suurta osaa en ollut ennen edes tavannut. Vakuutuin siitä, että tämä porukka saa kyllä lehden aikaiseksi. Ja vaikka olin nähnyt omaa tekstiäni painettuna jo monta kertaa, ensimmäinen Skrolli tuntui silti aivan erityiseltä.

Leipäpuuni on toisaalla, kuten muillakin Skrollin tekijöillä ainakin toistaiseksi. Lehteä tekemällä ei rikastu, eikä se ole tarkoituskaan. Itselleni kysymys on ennen kaikkea siitä, että haluan saada päässäni pyöriviä tarinoita paperille. Lehdelle taloudellinen riippumattomuus on toki tärkeää, jotta voimme jatkaa ilmestymistä. Vielä tärkeämpää on kuitenkin tehdä sellaista Skrollia, josta pitävät sekä me itse että lukijamme. Se, miten pitkälle tämä tarina jatkuu, riippuu ennen kaikkea teistä.

Vuoden kuluessa toimitusporukkaan on tullut lisää jäseniä, ja olemme myös löytäneet lukuisia uusia avustajia. Heidän asiantuntemuksensa avulla olemme voineet laajentaa käsittelemiemme aiheiden kirjoa huomattavasti. Osittain siksi käsillä oleva numero onkin tähänastisista ajankohtaisin. Mukana on reportaasit peräti kahdesta harrastetapahtumasta, minkä lisäksi puemme päähämme tuoreet Oculus Rift -virtuaalitodellisuuslasit.

Useimmat avustajista ovat itse ilmoittautuneet toimitukselle, ja tähän rohkaisemme myös jatkossa. Ensimmäistä kertaa voimme jopa maksaa kirjoituksista nimellisen palkkion. Jos koet, että sinulla olisi kerrottavaa Skrollin lukijoille, lähetä sähköpostia tai tule roikkumaan IRC-kanavalle. Aiempi kirjoituskokemus ei ole tarpeen – tärkeintä on, että sinulla on jotain sanottavaa.

Skrolli

Tietokonekulttuurin erikoislehti

Yhteydenotot toimitus@skrolli.fi
ircnet — #skrolli

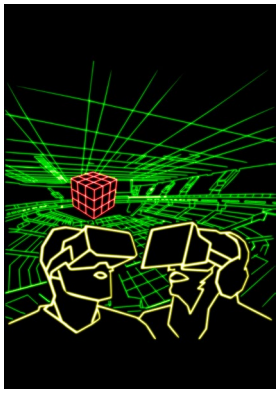
Päätoimittaja Ville-Matias Heikkilä
Toimituspäällikkö Toni Kuokkanen
Toimitussihteeri Ninnu Koskenalho
Taitto Risto Mäki-Petäys
Mediamyynti Jari Jaanto
Talous Anssi Kolehmainen

Muu toimitus Lauri Alanko, Mitol Berschewsky, Tapio Berschewsky, Mikko Heinonen, Jukka O. Kauppinen, Ronja Koistinen, Sade Kondelin, Juho Lehtinen, Elias Linjama, Oona Räisänen

Tämän numeron avustajat Anna Heinonen, Panu Kalliokoski, Aleksi Kinnunen, Antti Laaksonen, Waltteri Lahti, Santtu Pajukanta, Juho Pietarinen, Annika Piiroinen, Visa-Valtteri Pimiä, Tuomas Puikkonen, Rauli Puuperä, Manu Pärssinen, Ville Ranki, Mikko Rasa, Markku Reunanen, Miikka Saukko, Kalle Viiri, Jari Viitala

Julkaisija Alternative Party ry

Painopaikka Tammerprint, Tampere,
ISSN 2323-8992 (painettu)
ISSN 2323-900X (verkkojulkaisu)



Kannen kuva:
Ville-Matias Heikkilä

YMPÄRISTÖMERKKI
MILJÖMÄRKT
441 878
Painotuote

Testissä Oculus Rift
Lumemaailmojen sukelluslasit

Katsoimme keinotodellisuudeen houkuttavilla laseilla.

Teksti: Mikko Rasa Kuvat: Mikko Rasa, Wikimedia Commons-käyttäjä Sergey Galyonkin

Immersiivinen virtuaalitodellisuus on pelaajan nirvana ja science fictionin peruskauraa. Sen mahdollistavat laitteet ovat tehneet tietään kuluttajan luokse jo 1990-luvulta asti. Matkalla on ollut ongelmia, kuten huonot resoluutiot, epämukavuus ja hidas tai puuttuva pään asennon seuranta. Jos tekniset ominaisuudet onkin saatu kohdalleen, on hinta ollut liian kova kotikäyttäjälle.

Oculus VR-yhtiön virtuaalisilmikko Rift lupaa mullistaa pääsyn virtuaalimaailmoihin. Sen viimekesäisen Kickstarter-kampanjan suosio ylitti odotukset moninkertaisesti, ja toimitukset alkoivat tämän kevään aikana.

Maailma silmikkosilmin

Lupaukset ovat historian valossa suuria, mutta kun laitteen saa päähänsä, epäilykset haihtuvat. Syvyysvaikutelma on niin todentuntuinen, ettei siihen tajua kiinnittää huomiota, mutta jälkeenpäin tavallisella monitorilla esitetty 3D-grafiikka näyttää lattealta. Silmikon tarjoama näkymä kattaa lähes koko näkökentän, joskin reunoille jää laitteen rakenteesta johtuva musta alue. Näyttöpaneelin edessä olevien linssien ansiosta silmät eivät rasitu, ja mukana on vaihtolinssit likinäköisille.

Immersion täydentämiseksi kuvan on reagoitava pään liikkeisiin. Ihmisaivot olettavat ympäröivän maailman olevan pään liikkeistä riippumaton, joten kun käyttäjä kääntää päätään oikealle, on kuvaa liikutettava vastaavasti vasemmalle. Mikäli näin ei tapahdu, syntyy valheellinen mielikuva virtuaalimaailman kääntymisestä pään mukana. Reaktion on oltava riittävän nopea, tai aivot eivät osaa yhdistää pään kääntämistä ja katselusuunnan muutosta toisiinsa. Riftin asentosensori ja sitä lukeva kirjasto suoriutuvat tehtävästä mallikkaasti. Ympäriinsä katseleminen tuntuu luonnolliselta, eikä minimaalista viivettä huomaa, ellei sitä erikseen tarkkaile.

> *Oculus Rift lupaa mullistaa pääsyn lumemaailmaan.*

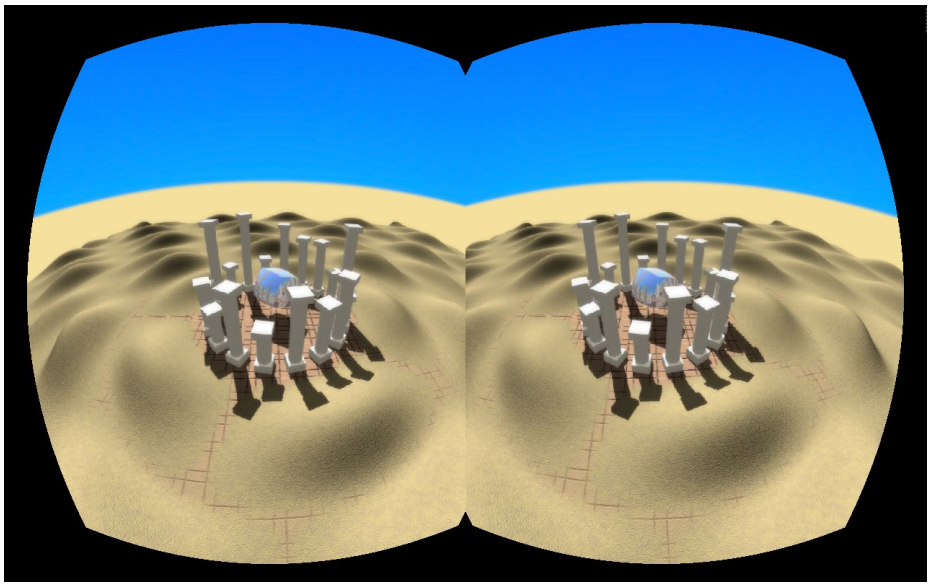
Aivan täydellinen laite ei ole. Kuvassa on voimakas hyttysverkkoilmiö, jossa pikselien väliset rajat erottuvat mustina viivoina. Tämä on ymmärrettävää, koska paneeli on vain muutaman sentin päässä silmistä. Myös kuvan tarkkuus on vaatimaton. Näytön 1280×800 pikselin tarkkuus on jaettu vaakasuunnassa kahtia, joten kumpikin silmä saa 640×800 pikselin näkymän. Tämä on kuitenkin vasta kehittäjille suunnattu versio. Myöhemmin ilmestyvään kuluttajaversioon on kaavailtu 1920×1080 pikselin paneelia, jossa myös pikselirajojen pitäisi olla kapeammat.

Kuulokkeita Riftissä ei ole, mutta sen muotoilu mahdollistaa omien isojenkin kuulokkeiden mukavan käytön. Pelien kontrollien tosin olisi syytä löytyä lihasmuistista, koska näppäimistöä ei silmikko päässä pysty tiirailemaan. Erillinen peliohjain onkin vr-kypärän kaveriksi oivallinen valinta.

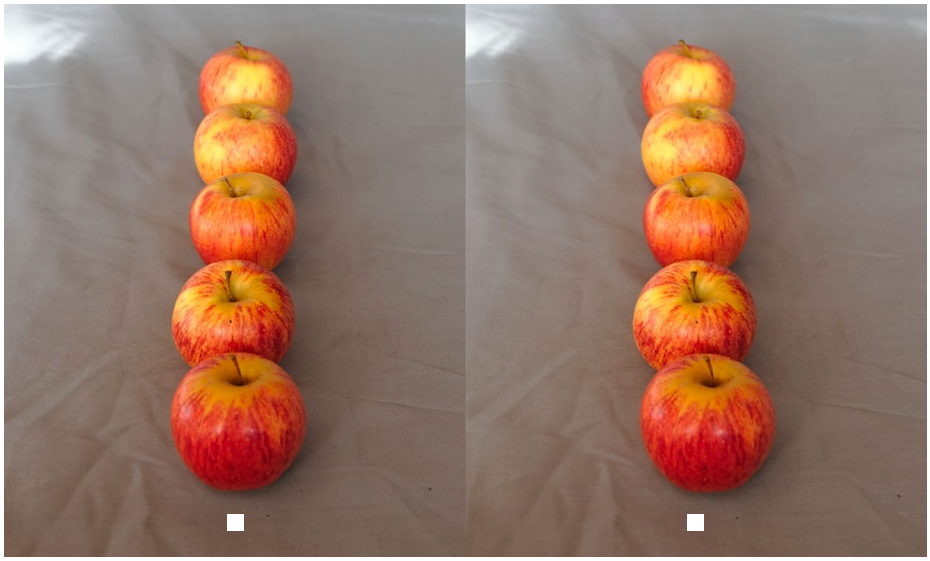
Pääseekö jo pelaamaan?

Koska kyseessä on uusi laite, joka ei vielä edes ole kuluttajamarkkinoilla, on pelituki varsin rajallinen. RiftEnabled-sivusto listaa noin 150 peliä ja ohjelmaa, joissa on natiivi Rift-tuki. Näistä suurin osa on teknologiademoja ja harrastajien kokeilumielessä tekemiä pikkupelejä, eikä isoja julkaisuja ole kuin muutama. Laajempaa tukea on epäilemättä luvassa lähitulevaisuudessa, sillä ainakin Valve ja Id Software ovat osoittaneet kiinnostusta laitteen tukemiseksi peleissään.

Valve on ensimmäisenä suurena yrityksenä lisännyt Rift-tuen Source-pelimoottoriinsa. Half-Life 2 Riftillä pelattuna on hieno kokemus. Pelaaja tuntee todella astuvansa Gordon Freemanin



Riftin näytölle piirtyvää kuvaa täytyy vääristää voimakkaasti, jotta se linssien läpi katsottuna näyttäisi oikealta. Vääristymän tarkoituksena on keskittää enemmän pikseleitä kuvan keskialueelle.



Sovita kuviot päällekkäin katsomalla kuvia silmät ristissä. Näin osat hyppäävät irti paperista.

saappaisiin. Kontrollit ovat hieman oudot, sillä hiirellä on vaakasuunnassa ruudun keskellä pieni kuollut alue, jolla kursoria voi liikuttaa vapaasti ennen kuin se alkaa kääntää pelihahmoa. Katseen suuntaa ja hahmon liikesuuntaa ei ole erotettu toisistaan, joten kävellessä ei voi katsella ympärilleen. Valve on ilmoittanut tämän olevan vielä keskeneräinen beta, ja Team Fortress 2:ssa tuen pitäisi olla valmiimpi.

Joihinkin peleihin saa lisättyä Rift-tuen avoimen lähdekoodin VireIO-ajurilla. Tämä kuitenkin jättää paljon toivomisen varaa. Näkymän parametrit on säädettävä kohdilleen käsin. Koska peli ei tiedä Riftistä ja sen aiheuttamasta vääristymästä mitään, kaksiulotteiset käyttöliittymäelementit eivät toimi kuten pitäisi. Esimerkiksi Portal 2:ssa valikko on hankalasti näkökentän vasemmassa laidassa. Pelin aikana tähtäin näkyy kahdessa eri paikassa, ja todellisuudessa laukaus osuu näiden väliin.

3D-grafiikkaa harrastavat bittinikkarit voivat luoda itse omia virtuaalimaailmoja. Unityssä ja Unreal Engine 3:ssa on valmis tuki Riftille, ja muidenkin grafiikkamoottorien käyttö onnistuu, jos näkee hieman vaivaa. Kehittäjäsivustolta voi ladata ilmaisen dokumentaation ja lähdekoodit, jotka sisältävät tarvittavat matemaattiset kaavat vääristymän kompensointiin. Virallinen ohjelmistonkehityspaketti sisältää myös pään asennon seuraamiseen tarvittavat rajapinnat ja algoritmit. Tuki löytyy Windowsin lisäksi Linuxille ja Macille.

Oculus Rift (kehitysversio)
Näytön tarkkuus: 1280x800 kuvapistettä
Näkökenttä: noin 110°
Pään asennon seuranta: integroitu
Kuulokkeet: ei ole
Hinta: 300 dollaria

Linkkejä:
http://www.oculusvr.com
http://www.riftenabled.com
http://www.vireio.com

Virtuaalitodellisuuden tulevaisuudennäkymät

Teksti: Visa-Valtteri Pimiä

Vihdoinkin! Oikeasti toimivien virtuaalitodellisuuslaitteiden aikakausi on koittanut. Olen haaveillut immersiivisestä näyttölaitteesta niin kauan kuin muistan. Oculus Rift ei kuitenkaan ole vielä kaikkien unelmien täyttymys. Seuraava suuri haaste ei ole näytön resoluution kasvattaminen, 360:n asteen juoksumattojen kehittäminen tai 3D-ohjainten suunnittelu. Todelliset puutteet ja mahdollisuudet piilevät ohjelmistossa.

Vieläkään ei oikeastaan tiedetä, millaiset kokemukset parhaiten soveltuvat virtuaalitodellisuuteen. Toistaiseksi ei ole kehitetty täysimittaista, suuren budjetin tai korkealentoisen konseptin immersiivistä kokemusta.Tarjolla on lähinnä puolihuolimattomasti tehtyjä konversioita vanhoista 3D-peleistä tai yksinkertaisia esimerkkiohjelmia laitteiden potentiaalista.

Kaikkein eniten odotan syvällisiä, juonellisia kokemuksia jotka käyttävät virtuaalitodellisuuslaitteistojen ainutlaatuisia ominaisuuksia hyödykseen. Miltä tuntuisi eläytyä elämään neliraajahalvaantuneena sänkypotilaana, joka pystyy kommunikoimaan vain päänsä liikkeillä? Millaisia olisivat virtuaalimaailmassa koetut elämykset esimerkiksi demotaiteen parissa?

Seuraan tätä kehitystä ja osallistun siihen etujoukoissa. Uskon sen olevan merkittävin askel pitkään aikaan videopelien ja muiden immersiivisten kokemusten kehityksessä kohti yhä kunnianhimoisempia ja uskaliaampia suuntia.

Jos jokin juonen syvyyttä sisältävissä peleissä on tärkeää, niin rooleihin ja tilanteisiin eläytyminen. Saavatko uudet keinomaailmat minut välittämään kuvitteellisista hahmoista entistä enemmän? Jos kokemus on aiempaa uskottavampi, vetoavatko tekoni voimakkaammin tunteisiin?

Virtuaalitodellisuus mahdollistaa kuvitteellisiin maailmoihin uppoutumisen ja niiden sisällä toimimisen ennennäkemättömillä tavoilla. Laitteiden hinnat ovat tavallisten ihmisten kustannettavissa. Itsenäisten kehittäjien luomien virtuaalikokemusten aika on koittanut!

Toivon, että näitä moderneja silmälappuportaaleja kokeilleet tunnistavat niiden valtavan potentiaalin ja keksivät, kuinka parhaiten valjastaa ne ainutlaatuisia mielikuvitusmatkoja ja mahtavia elämyksiä varten.

Jokohan tällä vuosikymmenellä?

Vaikkei kuusiirtokuntia ja ajatusohjauspantoja saataisikaan vielä lähivuosina, niin saataisiinko edes ne toimivat virtuaalikypärät? Niitäkin kun on kehitetty ja odotettu jo aika pitkään.

Teksti: Ville-Matias Heikkilä Kuvat: Wikimedia Commons -käyttäjät Evan-Amos, ThePassenger

Erilaisia lumetodellisuuksia on esiintynyt ihmisten kuvitelmissa jo ammoisista ajoista alkaen, eikä ihme — onhan esimerkiksi unia nähty jo miljoonia vuosia. Jopa käsitys fyysisen todellisuuden "virtuaalisuudesta" oli olemassa jo varhaisessa buddhalaisessa filosofiassa.

Teknisen virtuaalitodellisuuden historia voidaan aloittaa 1800-luvun stereoskoopeista eli stereokuvien katsomiseen tarkoitetuista kakkuloista. Stereovalokuvauksesta tuli 1850-luvulla oikea buumi, ja stereoskooppeja myytiin satojatuhansia. Myös sata vuotta myöhemmin muotiin tulleiden "3D"-elokuvien käyttämä sinipuna-anaglyfia keksittiin 1850-luvulla.

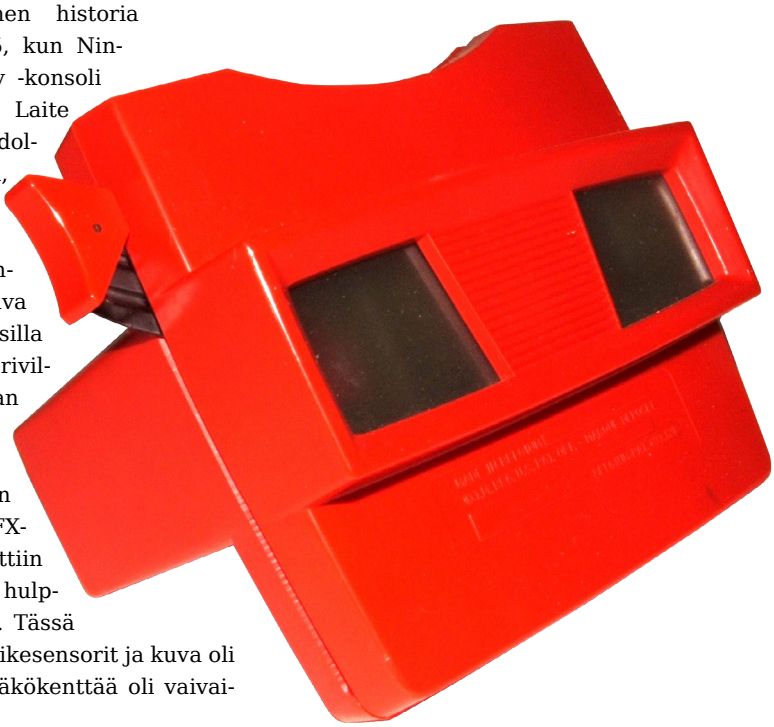
Vuonna 1961 rakennettu Sensorama oli kunniahimoinen varhainen lumetodellisuuskokeilu. Laitteessa istuvalle henkilölle esitettiin filmiltä laajakulmaista, värillistä stereoelokuvaa stereoäänillä, ja lisäksi hänelle tuotettiin liike-, tunto- ja hajuaistimuksia. Laite jäi kuitenkin kuriositeetiksi, sillä keksijä Morton Heilig ei saanut rahoitettua laajempaa tuotantoa.

TV-valmistaja Philco rakensi vuonna 1961 ensimmäisen kypäränäytön, joka mahdollisti todentuntuisen etäläsnäolon. Kypärän kuvan tuotti toisessa huoneessa ollut kamera, jota moottorit pyörittivät käyttäjän pään liikkeiden mukaan. Samantapaisen kypärän yhdisti tietokonegrafiikkaan vuonna 1968 amerikkalaistutkija Ivan Sutherland, jonka järjestelmä esitti käyttäjälle muutamasta viivasta koostunutta huonenäkymää.

Kypäränäyttöjä alettiin 70-luvulla kokeilla sotilastekniikassa. Näytöt edustivat lisättyä todellisuutta, jossa tietokonedataa heijastettiin osaksi lentäjän näkökenttää. 1980-luvulla tekniikka ei enää vaatinut miljoonien dollarien laskentarautaa, ja ensimmäiset graafisin työasemiin tarkoitetut järjestelmät pääsivät tällöin prototyyppiasteelle. Virtuaalitodellisuus-käsitteen vakiinnutti Jaron Lanier, joka perusti VPL Research -yhtiön vuonna 1985.

Ensimmäiset kaupalliset virtuaalisilmikot tulivat markkinoille 1989. Näitä olivat Eric Howlettin Cyberface ja VPL Researchin EyePhone. Cyberfacen kuva oli mustavalkoinen, mutta se tarjosi peräti 145 asteen näkökentän. Erikoisaloille tarkoitetut järjestelmät olivat kuitenkin aivan liian kalliita sohvankuluttajalle.

Kuluttajatason virtuaalitodellisuuden pettymyksentäyteinen historia alkoi vuonna 1995, kun Nintendon Virtual Boy -konsoli tuli markkinoille. Laite rakennettiin mahdollisimman halvalla, eikä sinä ollut esimerkiksi minkäänlaista liikkeentunnistusta. Kuva tuotettiin punaisilla lediriveillä ja pyörivillä peileillä. Hieman enemmän yritystä oli seuraavana vuonna myyntiin tulleessa Forte VFX-1:ssä, joka kytkettiin PC:hen ja maksoi hulppeat 1500 dollaria. Tässä juppilelussa oli jo liikesensorit ja kuva oli värillinen, mutta näkökenttää oli vaivaiset 45 astetta.



Vuosituhat vaihtui, mutta eri valmistajat jatkoivat samojen perusvirheiden tekemistä. Liiketunnistukseen ja latenssin minimointiin ei panostettu riittävästi, mikä aiheutti käyttäjille päänsärkyä. Kapeat, 30-50 asteen näkökentät puolestaan loivat illuusion ennemminkin pimeässä huoneessa leijuvasta näytöstä kuin joka suunnalta ympäröivästä keinotodellisuudesta.

Oculus Riftiä lienee suitsutettu etenkin siksi, että riittävän halvalla valmistettu laite on kerrankin onnistunut välttämään yleiset sudenkuopat. Vasta tulevat vuodet kuitenkin näyttävät, riittääkö tämä lumetodellisuuden läpilyöntiin, vai saammeko kenties odottaa vielä vuosikymmenen jos toisenkin.

Stereoskoopit ovat monelle lapsuudesta tuttuja. Ensimmäiset viihdekäyttöön myydyt virtuaalisilmikot eivät olleet juuri stereoskooppeja kummempia.

Maista uusi Kapsi!*

* Ei juotavaksi. Tarkempi sisällysluettelo www.kapsi.fi

Assembly 2013
Multimediaa mahan täydeltä!

Skrolli osallistui Assemblyihin lähes koko toimituksen voimin.

Teksti: Tapio Berschewsky, Ronja Koistinen, Toni Kuokkanen

Kuvat: Aleksi Kinnunen, Santtu Pajukanta

Assembly-tiimiltämme irtosi tapahtumasta useampikin henkilökohtainen tapahtumareppari. Emme halunneet editoida näitä yhteen, vaan julkaisemme kunkin osuuden omalla nimellään ja otsikollaan — kuten ne kirjoitettiinkin.

Moniääninen festariartikkelimme kertoo, millaista oli olla ensimmäistä kertaa uuden median edustajana oman alan tapahtumassa. Rakkaus oli molemminpuolista, ja näillä näkymin mikään ei pysäytä Skrollia olemasta paikalla myös ensivuonna.

Metronäyttömatopelin ständillämme tarjosi yleisölle Helsingin Hacklabin Konsta "sooda" Hölttä. Muut pelilaitteet saimme lainaksi keräilijäyhteisö Pelikonepeijooneilta. Kiitokset hyvistä juhlista kaikille osallistujille!

Kaukana skeneltä

Teksti: Tapio Berschewsky

Nyt se on sitten virallista. Assyjen lauantain kärkiohjelmanumeron paikan vei ensimmäistä kertaa ikinä peli. Asuksen StarCraft 2 -turnaus oli päälavalla tapahtuman paraatipaikalla lauantai-illassa — juuri siihen aikaan, jolloin demokompot ennen alkoivat. Pelit ovat nyt myös kesäassyjen ykkösjuttu.

Tosin eivät kyllä minulle. Olen pelaaja, mutta käyn Assyilla katsomassa demoja ja moikkaamassa kavereita. Ei voisi itse asiassa vähempää kiinnostaa peliurheilu, tai ylipäätään muiden lanittamisen katselu. Assyilla tai muualla. En tajua sitä, mutta se on oma ongelmani. Eikä se muutenkaan haittaa, sillä Assyt on vähemmän peliorientoituneellekin vieraalle täynnä viihdettä.

Takana on kymmenen vuotta juoksemista toisen median leivissä, nyt sai ottaa aika iisisti. Pöytävuorot jäivät pitkälti hoitamatta, onneksi muu tiimi paikkaili. Keskityin pitkästä aikaa itse tapahtumaan. Oli muuten aika hyvä sellainen.

Tuimia nuotteja

Jo torstaina kolisi ja kovaa. Maailmalla pelimusiikkikeikkoja on nähty enemmänkin, Suomessa 25-päisen orkesterin ja kuoron esittämät rakkaat teemat ovat aika harvinaista herkkua. Super Sound of Video Games hoiti homman kotiin kunnialla.

Käsittelyn saivat niin klassikot vuosien takaa kuin uudemmat suosikit Bubble Bobblesta Baldurs Gateen, Boulder Dashista Haloon. Huhhuh, että se pianolla soolotulkittu Giana Sistersin intromusa oli kova. Ihan hillittömän kova. Finaalina tarjoiltu Second Realityn läpisoitto toimi sekin, mutta jäi kesken. Jälkimmäisen puolen sovitusta odotellessa.

Perjantaina Kepes Mode onnistui tekemään erittäin vaisun vaikutuksen yleisöön. Jälkikäteen irc-kanavan logeja lukeneelle kävi selväksi, että keskimääräinen assykävijä ei erityisemmin kaivannut räppiä saliin. Mutta ketut siitä, itse pompin eturivissä koko keikan ajan. Ja junailinpa Jaffankin lavalle.

Kebu jäi välistä, se harmittaa jälkikäteen suuresti. Kiire vaan oli.

Koodin taikaa

Joka vuosi joku kompo on köyhä, tai useampikin. Oldskool oli torso, lyhytanimaatiot skippasin, demot jäivät vaisuiksi. Real Wildin kolme parasta vetosivat, vaan niin aina. Vähintään soodan metronäyttödemo next train takes no passengers tulee katsoa.

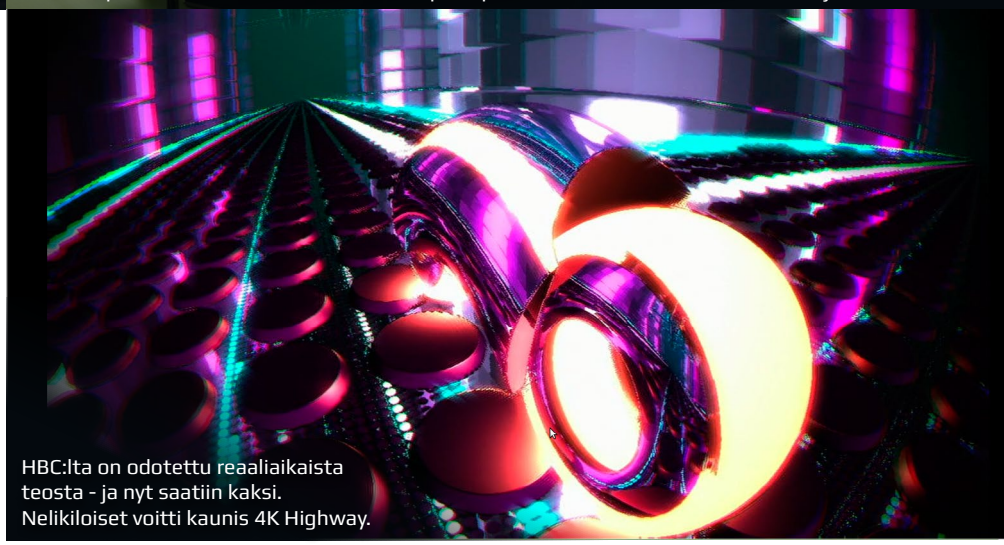
Nuori koodaaja väitti joskus kivenkovaan, että introja ei saa arvostella jos ei osaa ite. Hyvä jos edes katsoa, tälläinen likainen skenen ulkopuolinen epänörtti. Ei kiinnostanut, ja sanon nytkin, mistä tykkään. Nelikiloiset Highway 4K ja Yog-Sothoth olivat Assyjen screenin parhaat minuutit. Älkää jättäkö siihen, sillä molemmissa kokoluokissa oli paljon hyvää nähtävää. Kilotavun raytracer sai supistajalihaksen tutisemaan innosta.

Painavia sanoja

Vip-vieraana oli tällä kertaa hacktivisti Cory Doctorow. Hyvän ja YouTubestakin löytyvän How to Live in a World Made of Computers -puheen jälkeen ehdin istah-

Kunniavieras hacktivisti Cory Doctorow piti Assyjen kaikkien aikojen poliittisimman esityksen päälavalla. Katso se YouTubesta.

Super Sound of Video Games esitti upeaa pelimusiikkia klassisella orkesterilla ja kuorolla.



HBC:lta on odotettu reaaliaikaista teosta - ja nyt saatiin kaksi. Nelikiloiset voitti kaunis 4K Highway.

tamaan Coryn kanssa hetkeksi kahden kesken. Skrolli ei julkaise kovin lyhyeksi jäänyttä haastattelua, sillä itse puhe tyhjensi jo pankin.

Coryn terveiset lukijoille tosin välitämme: "Koko maailma rakentuu tietokoneista ja kaikki inhimillinen toiminta tehdään niiden avulla. Joka kerta kun näet tietokoneita koskevaa uutta lainsäädäntöä, kysy itseltäsi, hyväksyisitkö tämän lain koskemaan jokaista pienintäkin tekoa elämässäsi. Jos et hyväksy, pyri muuttamaan lakia."

18 vuoden välein

Teksti: Toni Kuokkanen

Kävin Assyilla ensimmäisen kerran 1995, ja nyt 18 vuotta myöhemmin olin siellä taas. Tästähän voisi vaikka tulla tapa. Skrollin ständillä oli kiva tunnelma, ja kävijöitä tuntui kiinnostavan pieni lehtemme.

Pelaajien määrä oli melkoinen, mutta hiljaisuus yllätti. Enää ei käydä desibelikisoja, mikä on kyllä ihan hyvä. Peltoreita ei tarvita. Edellisellä visiitillä konepaikoilla soi tauotta Josh Winkin Higher State of Consciousness, eivätkä volyymit varmastikaan olleet EU-normien mukaisia.

Olin tällä kertaa paikalla vain yhden päivän, joten Areenan nukkumatilat eivät tulleet tutuiksi. Myöskään kolajuomaövereille ei ollut tarvetta. Hienoisena parannuksena 90-lukuun huomasin, että ES:n nenäänpistävä käry peittää tehokkaasti alleen kolmen päivän ajan marinoituneen eau d'Assemblyn.

Tunnelma oli edelleen tallella, vaikka muutoksia on tapahtunut. Tuhansien kirkkaiden näyttöjen keskellä vanha dataaja tunsi olonsa etupäässä kotoisaksi. Kompoista en kerinnyt seuraamaan kuin grafiikan ja jokusen musakompon, joissa taso tuntui vaihtelevan reippaasti, kuten viime kerrallakin. 1995 mieleen jäi musakomposta Skavenin Catch That Goblin, joka kirvoitti yleisöstä melkoiset aplodit. Tällä kertaa suosikkia ei muodostunut.

Tapahtuma on selkeästi sivistynyt, eikä järjestyshäiriöitä juurikaan näy, vaikka ei mitään isompaa muistaakseni edelliselläkään kerralla ollut. Harvassa lienevät festarit, joilla on yli 3000 kävijää ja näin rauhallista. Dataporukka on selkeästi sivistynyttä väkeä.

Lähelle ydintä

Teksti: Ronja Koistinen

Olen käynyt Assemblyilla ensi kertaa 1999, jolloin fiilikset olivat vähintäänkin epätodelliset. Olin 14-vuotias enkä saanut vanhemmilta vielä lupaa yöpyä tapahtumapaikalla.

64 kilon intro oli älyttömän siisti asia. Muistan vieläkin PC:n 64k-kompon voittaneen Mewlersin Viagra-nimisen intron tapahtuman kohokohtana. Näytin koulun alettua sitä tietokoneluokassa kavereille, jotka jostain syystä eivät olleet yhtä vaikuttuneita.

Sittemmin olen käynyt Assemblyilla ainakin 2003, 2004 ja 2010. Demoskene ja tuttujen tapaaminen ovat aina olleet omat syyni tulla paikalle. En ole ikinä ollut etevä ohjelmoija, joten kosketus skeneen on ollut aika etäistä ja hankalaa.

Tänä vuonna fiilis oli ihan erilainen, kun olin paikalla Skrollin kanssa. Tuntui siltä, että pääsin tekemään lähempää tuttavuutta suomalaiseen skeneen ikään kuin takaoven kautta, leppoisalla metatasolla. Ständillä oleskellessa ehti jutella compoista tuttujen nimien kanssa. Kyllä journalismi avartaa!

Parhaat produt ja ohjelmat

Älä tyydy lukemaan, vaan katso itse parhaat Assy-produt tapahtuman hyvästä arkistopalvelusta Assembly Archive. Muista myös, että Assemblyt ovat vain se näkyvin Suomen tapahtuma. Kaunista ja päätä räjäyttävää materiaalia julkaistaan ympäri vuotta moninaisilla demopartyilla pitkin maailmaa ja Suomessakin partyja on useampia. Sukella siis syvemmälle skeneen. Demomaailman paras resurssi Pouet auttaa etsijän eteenpäin.

Syvempää skenetuntemusta ja lukemattomia upeita demoja:
http://pouet.net

Assyproduktiot 20 vuoden ajalta:
http://archive.assembly.org

Upeita syntetisaattoreitaan suvereenisti käsitellyt Kebu villitsi ihmiset perjantaina.

HUOM! Seuraava teksti saapui anonyymistä sähköpostiosoitteesta Skrollin toimitukselle. Kirjoittaja luovutti tekstin julkaistavaksi täysin oikeuksin. Julkaisemme tekstin lyhentämättömänä. Skrolli ei kannata päihtymistä tai lain rikkomista, ja esitetyt näkemykset ovat kirjoittajan omia.

Boozemblyt on peruttu

Sanovat Assemblya päihteettömäksi tapahtumaksi. Nuubit uskoo.

Teksti: Nano Muumi

Hartwall Areenan anniskelutilat ovat kiinni, eikä tuoppia saa vaikka sinnikkäästi anelisi. Ovella valvotaan, etteivät vieraat kanna pulloja tai lääkkeitä. Humaltuneen oloisia ei päästetä sisään, ja sisältä löytyneet passitetaan ulos. Energiajuomaa kovemmat mömmöt jäävät takavarikkoon.

Iltaisin järjestyksenvalvojat parveilevat Boozembly-reitin päässä. He tarkkailevat missä kunnossa aikuisviihteelle vaihtaneet kävelevät takaisin kohti Areenaa. Tuttuja moikkaillaan, ja molemmin puolin tiedetään mistä on kyse. Tässä katsotaan, että riehujat ja hoipertelijat päätyvät toiseen osoitteeseen.

Samaan aikaan reitin puolivälissä ujoimmat kulkijat seisahtuvat kittaamaan viimeisen kaljan pohjat, jotta tölkin voi jättää tarpeeksi kauas näkyviltä. Assyt ovat päihteettömät periaatteella "älä kysy, älä kerro". Pieniä sieviä nauttineet tai siltä näyttävät pääsevät takaisin sisään, läpi seulasta.

Boozemblyllä ei virallisesti ole järjestäjiä. Joku ilmeisesti silti hakee sille joka vuosi salaa hupiluvan, sillä poliiseja ei koskaan näy paikalla. Alue myös siivotaan huolellisesti joka ikinen vuosi. Irkissä muistetaan jekuttaa nuorempia. Joka vuosi Boozet on peruttu, tai siirretty, tai jotain.

Passaa vasemmalle

Pidemmällä, metsän puolella, musa soi ja ihmiset bilettävät. Reippaasti yli sata iloista päätä erilaisissa sekavuuden tasoissa ja tiloissa jakavat yhteistä kesäyötä. Suurimmalla osalla juoma kädessä ja joitain takana, osa tukevassakin humalassa.

Niin kuin kaikissa isoissa tapahtumissa, siellä täällä vilahtelee jointteja. Tutun tai tuntemattoman kotikasvattama kukka tuoksuu, suuren tekniikkayrityksen lahjoittama kalja maistuu, ja tunnelma on katossa. Vanhat ystävät tapaavat, monet ainoan kerran vuodessa.

Yön mittaan väki vaihtuu tiuhasti. Viinapiru kaataa osan nukkumaan, ja osa siirtyy viettämään iltaa muualla. Jotkut siirtyvät takaisin Assyille, ja joitakin kiinnostaa Hashemblyn puoli.

Pilvenpolttelukallion rauhallisempi porukka puhuu rennosti ja huomattavasti matalammalla volyymilla kuin isompien epävirallisten bileiden juomaveikot. Vähän sivummassa eivät edes häiritse niitä muutamaa ahdistuneempaa boozekävijää, joille kukka on murhaa.

Paperit suuhun

Tietokonekulttuuri ei ole mikään poikkeus siitä, että ihmisten juhlintaan liittyy usein päihtyminen. Pinnan alla kuplii vähän vahvempiakin aineita. Psykedeelit ja stimulantit ovat läsnä, vaikka marginaalin asemassa. Pienen piirin omasta kivasta ei juurikaan näe jälkeä, jos ei osaa katsoa oikein. Boozet ovat siisti tapahtuma, ja ylilyöntejä ei juuri näy.

Assyjen päihteettömyyden verho pysyy uskottavana, vaikka viereisessä puistossa ihmiset kiskovat päänsä täyteen yhtä sun toistakin. Se kertoo vain siitä, miten hyvin demoskene ja sen hengaajat hoitavat hommansa. Täällä päihdytään, mutta pääosin taidolla.

Jos joku sattuukin vetämään pleksit, ja pyrkii takaisin Assyille, järkkärit ohjaavat sekoajan kohti Pasilan asemaa. Hyvä niin — tapahtuman puhdas maine on se, minkä takia vanhemmat uskaltavat päästää lapsensa paikan päälle valvomatta. Ilman keskivertokävijää ei Assyja olisikaan.

Koodia pennuille

Ninnu Koskenalho

Peruskoulussa ei opeteta ohjelmointia. Tietotekniikan kielen puutteellinen ymmärrys on aukko yleissivistyksessä.

Osallistuimme Skrollin kanssa Assemblyille. Visiitti oli minulle ensimmäinen sitten 2000-luvun alun. Olin tuolloin rampannut tapahtumassa vuodesta -94 tarkkaillen sen hidasta muodonmuutosta yhdessä tietotekniikan viihdekäytön yleistymisen kanssa. Jälleennäkeminen herätti ajatuksia suomalaisesta dataipanaskenestä ja tietotekniikan käyttötavoista.

Kuin pyy maailmanlopun edellä

Areenan asukkien keskimääräinen koko oli kymmenessä vuodessa pienentynyt. Viikonlopun kuikuilun perusteella suurin osa assykävijöistä oli pääasiassa ottamassa osaa isoille laneille.

Seitsemänvuotias minä istui 80-luvulla espoolaisessa lähiökodissa seikkailemassa pelimaailmoissa itsekseen. Pihan pojista ei tyttöbakteereja pelkäävinä ollut pelikavereiksi, eivätkä tytöt vielä kasarilla ymmärtäneet hauskan päälle.

2010-luvulla kaikki pelaavat. Seitsenvuotiaat siskonpoikani rakentavat Minecrafttiin uskomattoman eeppisiä todellisuuksia. Löydän itseni tivaamasta rakennuspiirrosten patentinhaltijan nimeä ja vaatimasta taidonnäytteiden toistoa valvotuissa olosuhteissa.

Minkä tahansa hupikäytettävän hilavitkuttimen saumaton hallinta uppoaa tenavien selkäytimeen paljon omaani nopeammin. Etumatkaksi ei riitä elämänmittainen altistus nappien rytmikkäälle painelulle.

Tätä kai tarkoitetaan diginatiiveilla. Mikä siis olisikaan sopivampaa, kuin että tietokonekulttuurin keskelle syntynyt sukupolvi jo nuorella iällä kansoittaa sen omia tapahtumia.

Skrollin flaikkuja nuorten näppiksille jakaessa mietin, moniko mahtaa olla kiinnostunut siitä, mitä pelialustan pinnan alla tapahtuu. Pureeko itsetekemisen promoaminen kymppiluvun nuorisoon? Kiinnostaako haksorointi, napostelleeko koodaus?

2000-luvun kansalaistaidot

Tietokonekulttuurista puhuminen on sikäli hupaisaa, että termi kattaa joka päivä suuremman osan kaikkea inhimillistä kokemusta. Tietokoneista on ihmisten maailma tehty, ja niillä sitä pyöritetään. Siihen nähden on yllättävää, että peruskoulun opetusohjelmaan kuuluviin pakollisiin, yleissivistäviksi katsottaviin oppiaiheisiin ei kuulu tietojenkäsittelyn ymmärryksen perusteita.

Matemaattisten aineiden opettajien liitto MAOL ry:n julkaisu “Mitä peruskoulunsa päättävän oppilaan tulisi tietää tietotekniikasta” listaa tavoitellun yhteisen taitopohjan. Siihen kuuluvat käyttöliittymän perushallinta, tekstinkäsittely ja taulukkolaskenta, tietokantapalvelut, piirto-ohjelma sekä sähköpostin ja hakukoneen käyttö.

> *Virossa ohjelmointia opetetaan 7-vuotiaille.*

Ilman näitä taitoja on maailmassa jo nyt surkeaa navigoida, saatika nykylasten aikuisuudessa. Vaan kuinka ihmeessä tietojärjestelmien toimintalogiikka ja ohjelmoinnin perusteet eivät ole listalla? Niiden opettaminen on nykypäivänä vähintään yhtä tärkeää kuin vaikkapa ruotsin kielen, tuon ikuisen kiistakapulan.

Toinen kotimainen kieli on jokaiselle suomalaiselle kielilain nojalla pakollinen opiskeltava. Kielilaki takaa kansalaiselle oikeuden viranomaispalveluihin kummalla tahansa kotimaisella kielellä. Valtiolla on velvollisuus taata oikeuden toteutuminen kouluttamalla ruotsin kielen osaajia.

Valtiolla olisi erittäin pätevä syy haluta niin ikään taata tulevaisuuden yhteiskunnalle reippaat rivit tietoteknisten palveluiden älykkäitä, kykeneviä tuottajia.

Kouri robotin aivoja rohkeasti

Mitä useampi ihminen osaa ja uskaltaa kurkistaa maailmaa kansoittavien koneiden pinnan alle, sitä vähemmän meidän tarvitsee tulevaisuudessa naamakämmentää kaikkia koskevien tietojärjestelmien uudistuksille.

Ei kaikista tule isona koodereita ja haksoreita. Ei kaikista tule matemaatikoitakaan, mutta matematiikan perusopetuksen tärkeys on täysin yleisymmärrettävä seikka.

Koululaitoksen tehtävä on koulia lapsista pärjääviä kansalaisia. Tietoteknistyvän mediayhteiskunnan tärkeimpiä kansalaistaitoja ovat medialukutaito ja tietotekninen ymmärrys. Kumpikin vaatii kyseenalaistamista, asioiden osiin purkamista ja uudelleen kasaamista.

Peruskoulun on syytä avata lapsille konepellin kansi. Jokainen ipana ansaitsee saada hyvät matkaevät kasvaa tietokonekulttuurin saralla enemmän kuin kuluttajaksi.

Virossa alettiin viime vuonna opettaa ohjelmoinnin perusteita kaikille koululaisille 7-vuotiaasta alkaen. Seurataan esimerkkiä!

Havainnoi. Hakkeroi. Rakenna.

OHM2013 järjestettiin Alankomaissa Oudkarspelissa Alkmaarin lähellä. Loppuunmyytyyn tapahtumaan osallistui 3000 ihmistä ympäri maailmaa.

Teksti: Rauli Puuperä Kuvat: Miikka Saukko

Hollannissa on järjestetty isoja kansainvälisiä hakkerikonferensseja vuodesta 1989. Neljän vuoden välein järjestettävät tapahtumat keräävät osallistujia ympäri maailmaa. Tänä vuonna tapahtuman nimi oli "Observe. Hack. Make." - Havainnoi. Hakkeroi. Rakenna.

Tapahtuman esitykset oli myös jaettu näihin kolmeen teemaan.

Keskieurooppalaiset hakkerileirit ovat erikoisia tapahtumia. OHM2013 on ikäänkuin rokkifestivaali, jossa musiikki on korvattu konferenssiesityksillä. Tämäkään kuvaus ei täysin onnistu kertomaan, mistä oikeastaan on kyse.

Hakkerin paratiisi

OHM2013 -tapahtumassa nörttikulttuuri tuli esiin mitä moninaisimmin tavoin. Leirialuetta kierrellessä voi nähdä Tesla-käämejä, erilaisia lennokkeja, kaikenlaisia elektroniikkaprojekteja, tiirikointia, radioamatöörilaitteistoa, 3d-printtereitä, laserleikkureita ja paljon, paljon muuta. Yksi alue oli kokonaan omistettu retropeleille. Pelialueella pääsi palaamaan omaan (ja myös vanhempiensa) nuoruuteen vanhojen flippereiden ja kolikkopelien parissa.

Tällaisen valtavan kokonaisuuden pystyttäminen vaatii paljon resursseja. Valtava urakka järjestetään täysin vapaaehtoisvoimin. Pääjärjestäjien lisäksi leirillä tarvitaan paljon vapaaehtoisia. Osallistuminen on helppoa. Tapahtumaan myytävä "peruslippu" on nimeltään volunteer ticket, eli kaikkien toivotaan kantavan kortensa kekoon.

Tämän vuoden näkyvin teema olivat tietovuodot. Puhujien joukossa oli monia nimekkäitä tietovuotajia Yhdysvalloista ja Isosta-Britanniasta, jotka kertoivat omista kokemuksistaan vuotajina olemisesta. Edward Snowden mainittiin monessa esityksessä.

Leirin ehdottomasti kuuluisin puhuja, Julian Assange, oli vielä neljä vuotta sitten paikan päällä esiintymässä. Tänä vuonna hän hoiti oman esityksensä videopuhelun avulla Ecuadorin Lontoon suurlähetystöstä käsin.

Tietovuotoaiheiden lisäksi nelipäiväisessä tapahtumassa oli esityksiä muun muassa politiikasta ja erilaisten laitteiden tekemisestä. Nimekkäin esitelty projekti oli Orvillecopter — rakkaasta lemmikkikissasta tehty nelikopteri. Orvillecopterin tekijät esittelivät myös isompaa aikaansaannostaan strutsikopteria. Myös monet erilaiset tietoturva-aiheet olivat esillä.

Pari piuhaa ristissä

Vertaaminen rokkifestariin on ainakin puitteiden puolesta vähättelyä. Rokkifestareilla perusinfrastruktuuri, kuten vessat ja suihkut, tuottavat työtä järjestäjille. Hakkerileirillä näiden lisäksi pitää olla tietysti myös sähköä ja internet-yhteys. Verkkotiimin jäsenet olivat vetäneet kilometrikaupalla valokaapelia ympäri leirintäaluetta, jotta jokaisella olisi pääsy nettiin.

Verkkoyhteydet hoituivat datavessojen (Datenklo) avulla. Bajamajoista oli tehty yhdyspisteitä, joista jokaiseen telttaan saatiin vedettyä verkkopiuha.

Wikileaksin nimekäs pääjehu Julian Assange piti puheen OHMeilla, tosin etänä Ecuadorista.

OHMeilla oli nettikuuluisuutta jos jonkinlaista, kuten Orvillecopter. Kiistellyssä projektissa rakas vanha lemmikki on kuoleman jälkeen muutettu nelikopteriksi. Rakkautta vai häväistys?

Elektronisen sodankäynnin kylässä nähtiin myös tämä restauroitu Vampire V2 75FL93. Radiolaitteet on saatu toimiviksi, mutta joitain pieniä puutteita vielä löytyy.

Jokaisessa datavessassa oli myös WLAN-tukiasema, ja langaton verkko toimi tänä vuonna paremmin kuin aikaisemmissa tapahtumissa. Jos nyt hakkeritapahtumassa ylipäätään uskaltaa siirtää tietoa langattomasti.

Sähköntarvetta tyydyttämään alueelle hoidettiin useita agregaatteja. Paikan päällä oli myös peräti kaksi omaa GSM-verkkoa ja DECT-verkko.

Kylittäin kavereita

Tapahtuman leirintäalue oli monen kirjavaa. Ihmiset olivat organisoituneet erilaisiksi "kyliksi". Kylä voi muodostua kansalaisuuden, aatteen, ohjelman, jonkun hackerspacen tai kaveriporukan ympärille.

Osa kylistä oli todella panostanut äänentoistoon ja valoilotteluihin. Hollantilaisen Hack42-hackerspacen kylä kuului näyttävimpiin. Neljään rakennustelineistä koottuun torniin oli viritetty useita valoja, discopalloja ja liekinheitin. Tätä kelpasi katsella Alankomaiden hämärtyvässä yössä.

Paikalla oli vajaa parikymmentä suomalaista. Osa suomalaisista majaili Finnish Embassy Perkele -kylässä, joka suomalaisittain kuului leirin vaatimattomampiin. Kylän asukkaat olivat saapuneet lentäen ja lentokoneiden matkatavaroiden painorajoitukset hankaloittivat kylän kasaamista.

Osallistujia oli Helsingistä, Tampereelta ja Oulusta. Suurin osa suomalaisista toimi tietoturva-alalla. Paikalla oli myös hackerspace-aktiiveja. Jotkut olivat työmatkalla, toiset lomalla, mutta kaikki teltoissa yhtä lailla.

Vuoroin vieraissa

OHM2013 ja vastaavat hakkerileirit ovat lapsiystävällisiä tapahtumia, ja perheen voi ottaa mukaan. Lapsille on tarjolla jonkin verran järjestettyä ohjelmaa — nämä voivat esimerkiksi päästä juottamaan, jos haluavat vaikkapa opetella ledivilkkujen väkertämistä. Perheloma hakkerileirillä varmasti avartaa nuorten käsityksiä siitä, mitä kaikkea maailmassa ylipäätään voi tehdä ja saada aikaiseksi.

Myös saksalainen Chaos Computer Club (CCC) järjestää vastaavaa tapahtumaa neljän vuoden välein. OHMin jatkoa Hollannissa saa siis odottaa vielä neljä vuotta, mutta seuraava mahdollisuus päästä kokemaan kansainvälisen suuren hakkerileirin humua on jo 2015 Saksassa, kun seuraava Chaos Communication Camp järjestetään.

Suomalainen leirillä

Vähä-Sipilä kutsuu pelaajat paikalle

Tietoturvan parissa työskentelevä Antti Vähä-Sipilä on käynyt viimeisen kahdeksan vuoden ajan hakkerileireillä. "Nämä ovat olleet hauskoja tilaisuuksia. Merkkasin OHM2013:n kalenteriin heti, kun siitä tuli tieto, ja ostin liput ensimmäisenä päivänä". Vähä-Sipilän mielestä tämänvuotinen leiri oli vähän erilainen kuin aiemmat. Tietovuotokeskustelu vei huomiota ihmisten jokapäiväisestä rakentelusta ja nysväämisestä, jotka ovat hänelle tärkein syy tulla paikalle. "Poliittiset keskustelut ovat tärkeitä, mutta niitä varten ei tarvitse lähteä telttaleirille", Vähä-Sipilä kommentoi.

Vähä-Sipilän mielestä leirien esitykset ovat laadukkaita. Niitä pitävät samat tyypit, jotka käyvät puhumassa "oikeissakin" tietoturvakonferensseissa. Laatu on vaihtelevaa, koska ohjelmaa on määrällisesti enemmän, ja aiheet ovat laajempia. Esitykset ovat kuitenkin samassa sarjassa, koska ne käyvät läpi hyväksyntäprosessin aivan kuten tavallisissakin konferensseissa. Kaikki esitykset arvioidaan etukäteen riittävän hyviksi, eikä mätiä mahdu mukaan.

Vähä-Sipilän mielestä OHM2013-tapahtuman kaltaista hakkerileiriä kannattaa ehdottomasti kokeilla ainakin kerran. "Tänne kannattaa raahata semmoset tyypit, jotka on menossa Assyille vain pelaamaan. Jos tietokoneen käyttö on vain pelaamista, siitä jää paljon uupumaan. Jos joku semmonen tyyppi pelastetaan täysipainoisen tietokoneharrastuksen pariin — saisi vaikka kipinän tämmösen tapahtuman kautta — niin se olisi hyvä", Vähä-Sipilä veistelee.

Nauhoitettuja esityksiä pääsee katsomaan osoitteesta http://wipkip.nikhef.nl/events/OHM/video/

QML osa 2
Yhteys C++ -maailmaan

QML-käyttöliittymien tekoon tutustuttiin Skrollin ykkösnumerossa. Nyt teemme liittymän C++-ohjelmalle.

Teksti: Ville Ranki Kuva: Jari Viitala

Skrollin QML-kurssin kakkososassa selvitämme, kuinka helposti QML:llä tehdyn ohjelman saa tekemään yhteistyötä C++:n kanssa. Kuvitteellisessa esimerkkitilanteessamme meillä on Arduinolla toteutettu saunavahti, joka osaa mitata saunan lämpötilaa ja käynnistää tai sammuttaa saunan. Lisäksi haluamme näyttää lämpötilahistoriaa viiden tunnin ajalta. Arduinon juttelu sarjaportin yli on toteutettu aiemmin C++:lla, mutta nyt haluamme tehdä sille kosketusnäytöllä käytettävän käyttöliittymän. Perinteisesti ohjelmointikielten välinen tiedonvälitys on työlästä, mutta QML:n ja C++:n tapauksessa tarvittavan liimakoodin määrä jää hyvin pieneksi.

Aloitamme luomalla uuden projektin ja siihen SaunaControl-nimisen luokan, joka toimii rajapintana QML- ja C++-maailmojen välillä. Käynnistä Qt Creator ja luo uusi Qt Quick 2 -projekti. Klikkaa projektia oikealla napilla ja valitse "Add New", "C++ Class". Anna luokan nimeksi SaunaControl ja kantaluokaksi (base class) "QObject". QObject on Qt:n luokka, jonka perimällä mikä tahansa luokka saa supervoimia kuten signaalit ja slotit, ominaisuudet (properties) ja muistinhallintaa helpottavia asioita. Signaaleista ja sloteista voi lukea tarkemmin linkistä [1] ja ominaisuuksista linkistä [2].

Kirjoita listauksen 1 koodi tiedostoon saunacontrol.h ja listauksen 2 koodi saunacontrol.cpp.

```
#ifndef SAUNACONTROL_H
#define SAUNACONTROL_H
#include <QObject>

class SaunaControl : public QObject {
    Q_OBJECT
    Q_PROPERTY(int temp READ temp WRITE
setTemp NOTIFY tempChanged)
public:
    explicit SaunaControl();
    void setTemp(int newTemp);
    int temp();
signals:
    void tempChanged();
public slots:
    void setSaunaPower(bool power);
private:
    int temperature;
};
#endif // SAUNACONTROL_H
```

Listaus 1: saunacontrol.h

```
#include "saunacontrol.h"
#include <QDebug>

SaunaControl::SaunaControl() : QObject(),
temperature(0) {}

void SaunaControl::setTemp(int newTemp) {
    temperature = newTemp;
    emit tempChanged();
}
int SaunaControl::temp() {
    return temperature;
}
void SaunaControl::setSaunaPower(bool power)
{
    qDebug() << Q_FUNC_INFO << power;
}
```

Listaus 2: saunacontrol.cpp

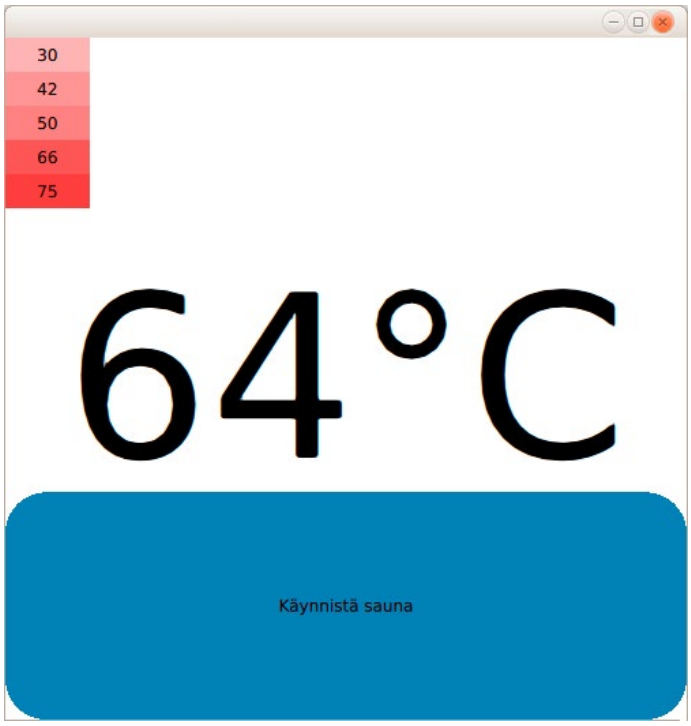
SaunaControl toimii tiedonvälittäjänä C++- ja QML-ohjelmien välillä. Luokassa on rakentaja, metodi näytetyn lämpötilan asettamiseksi ja palauttamiseksi, signaali joka kertoo lämpötilan muuttuneen sekä slotti saunan virran kytkemiseksi päälle tai pois. Q_PROPERTY-rivillä kerrotaan, että meillä on ominaisuus temp, sen lukumetodi, kirjoitusmetodi ja signaali, joka kertoo lämpötilan muuttuneen. Tämän ominaisuusmäärityksen avulla lämpötilamuuttujaan päästään käsiksi QML-puolelta suoraan. Myös lämpötilan muuttaminen QML:stä toimii, vaikka sitä ei tässä esimerkissä tarvitakaan.

SetTemp-metodia kutsutaan main.cpp:stä. Sen toteutus emittoi signaalin tempChanged(), jonka avulla QML-puoli (ja mahdolliset muut tätä signaalia kuuntelevat) tajuavat päivittää arvon.

SetSaunaPower-metodia taas kutsutaan QML-koodista, kun käyttäjä painaa painiketta. Tässä esimerkissä se tulostaa saamansa parametrin.

Kirjoita listaus 3:n sisältö Qt Creatorin luomaan main.qml-tiedostoon ja listaus 4:n sisältö main.cpp-tiedostoon.

Tarkastellaan main.cpp:tä. Sen alussa luodaan lämpötilahistoriaa kuvaava tempHistoryData-niminen merkkijonolista, johon laitetaan muutama kiinteä arvo. Merkkijonolista asetetaan "tempHistory"-nimiseen kontekstin ominaisuuteen, jolloin se näkyy globaalisti QML-puolella. Tiedoston main.qml loppupuolella on ListView-olio, joka osaa näyttää tietomallin. Tietomalli asetetaan ListView'n model-ominaisuudeksi. Delegate-ominaisuudeksi taas asetetaan QML-elementti, joka



Kuvakaappaus esimerkkisovelluksesta.

luodaan jokaiselle tietomallin alkiolle, tässä tapauksessa siis lämpötila-arvolle. Kyseinen arvo näkyy delegaatin modelData-ominaisuutena. Tässä esimerkissä käytettiin yksinkertaisuuden vuoksi valmista merkkijonolistaa, mutta delegaatit pystyvät käyttämään myös mitä tahansa sopivasti tehtyä QObjectista perittyä luokkaa.

Delegaattien avulla saadaan esimerkiksi räiskintäpelissä kuvattua pelihahmot C++-olioina, joiden ominaisuudet päivittyvät itsestään QML-puolelle piirtoa varten. Ominaisuuksia ei näin tarvitse päivitellä käsin pelin päässilmukasta.

Lämpöhistorian jälkeen main.cpp:ssä luodaan SaunaControl-instanssi ja asetetaan se juurikontekstille nimellä "saunaControl". Nyt SaunaControl:n slotteja voidaan kutsua ja sen temp-ominaisuus voidaan lukea ja kirjoittaa QML-koodista. Lopuksi asetetaan vielä lämpötila arvoon 64 astetta.

Main.qml:n alussa oleva Text-elementti näyttää isolla näkyvän tämän hetkisen lämpötilan. Siinä viitataan suoraan SaunaControl:n temp-ominaisuuteen.

Sauna kytketään päälle tai pois main.qml:n powerButton-napilla. Huomaa MouseArea-elementin onClicked-käsittelijä, joka kutsuu saunaControl:n setSaunaPower()-metodia aiheuttaen tulostuksen C++-puolella.

Qt tarjoaa esiteltyjen lisäksi myös muita tapoja välittää tietoa C++:n ja QML:n välillä. Toinen ihan hyvä toteutustapa olisi ollut luoda oma QML-elementtityyppi jonka toteutus on C++:aa. Tämäkin on yllättävän helppoa, riittää että oma luokka perii QtQuickItem:n ja rekisteröidään qmlRegisterType()-metodilla näkymään QML-puolella. Yksi, ei ehkä niin siisti tapa on suorat funktiokutsut. Esimerkiksi lämpötilan asetuksen olisi voinut tehdä myös tekemällä QML-puolelle JavaScript-funktio, jota kutsuttaisiin C++-puolelta QMetaObject::invokeMethod()-metodilla.

Lisää aiheesta voit lukea linkistä [3].

```
import QtQuick 2.0

Rectangle {
 width: 500
 height: 500
 Text {
  text: saunaControl.temp + "°C"
  font.pixelSize: parent.height/3
  anchors.centerIn: parent
 }
 Rectangle {
  id: powerButton
  property bool saunaPower: false
  color: saunaPower ? "salmon" : "steelblue"
  radius: 30
  width: parent.width
  height: parent.height/3
  anchors.bottom: parent.bottom
  Text {
   anchors.centerIn: parent
   text: powerButton.saunaPower ? "Sammuta sauna" : "Käynnistä sauna"
  }
  MouseArea {
   anchors.fill: parent
   onClicked: {
    powerButton.saunaPower = !powerButton.saunaPower
    saunaControl.setSaunaPower(powerButton.saunaPower)
   }
  }
 }
 ListView {
  width: parent.width/8; height: parent.height
  model: tempHistory
  delegate: Item {
   height: 25
   width: parent.width
   Rectangle {
    color: "red"
    opacity: parseInt(modelData)/100
    anchors.fill: parent
   }
   Text { text: modelData + "°C"; anchors.centerIn: parent }
  }
 }
}
```

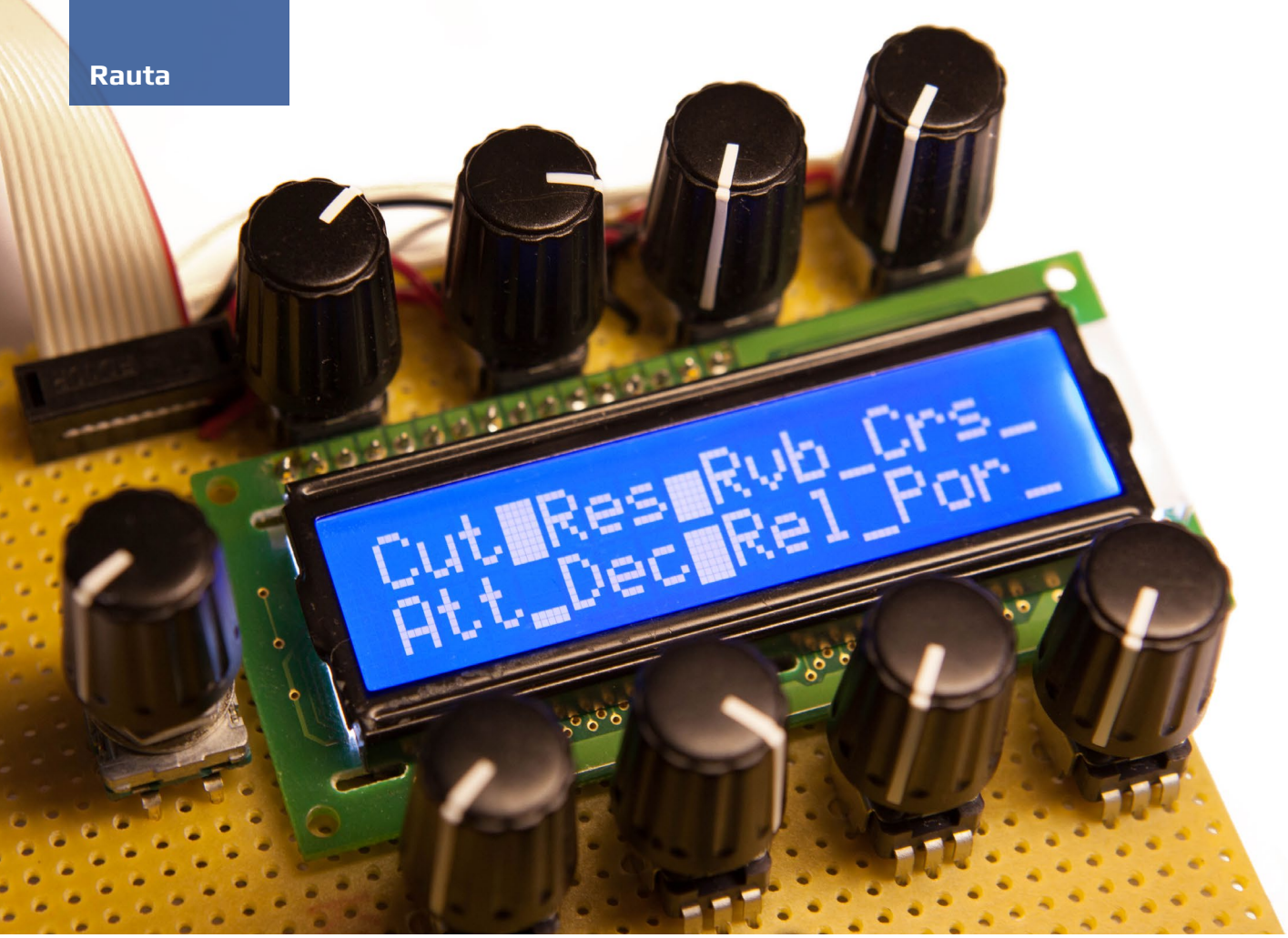
Listaus 3: main.qml

```
#include <QtGui/QGuiApplication>
#include <QQmlContext>
#include "qtquick2applicationviewer.h"
#include "saunacontrol.h"

int main(int argc, char *argv[]) {
    QGuiApplication app(argc, argv);
    QtQuick2ApplicationViewer viewer;
    QStringList tempHistoryData;
    tempHistoryData << "30" << "42" << "50" << "66" << "75";
    viewer.rootContext()->setContextProperty("tempHistory",
            QVariant::fromValue(tempHistoryData));
    SaunaControl saunaControl;
    viewer.rootContext()->setContextProperty("saunaControl",
            &saunaControl);
    viewer.setMainQmlFile(
            QStringLiteral("qml/skrolli-sauna/main.qml"));
    viewer.showExpanded();
    saunaControl.setTemp(64);
    return app.exec();
}
```

Listaus 4: main.cpp

Linkit:

[1] http://qt-project.org/doc/qt-5.0/qtcore/signalsandslots.html
[2] http://qt-project.org/doc/qt-5.0/qtcore/properties.html#qt-s-property-system
[3] http://qt-project.org/doc/qt-5.0/qtqml/qtqml-cppintegration-interactqmlfromcpp.html



Arduino hallintaan

Muokattava käyttöliittymä mikrokontrollerille

Säädä arduinosi toimintaa ilman tietokonetta kätevällä käyttöliittymällä.

Teksti: Annika Piiroinen
Projektin idea ja toteutus: Risto Mäki-Petäys
Kuvat: Risto Mäki-Petäys

Mikrokontrollerilla voi tehdä paljon muutakin kuin vilkuttaa ledejä tai tehdä saunamittareita. Niiden ei tarvitse olla vain kojeita, jotka valmistumisensa jälkeen suorittavat uskollisesti aina samaa toimintoa hamaan loppuun asti – tai kunnes niille ladataan uusi koodi. Tämä artikkeli on suunnattu niille, joilla on jo kokemusta Arduinolle ohjelmoimisesta, mutta tahtovat tehdä sille jonkin laajemman projektin.

Muokattava käyttöliittymä tarkoittaa sitä, että laitteen toimintaa voi ohjata käyttöliittymän kautta, mutta myös käyttöliittymän itsensä toimintaa voi muokata. Nappulat, jotka nyt säätelevät joitain tiettyjä arvoja, voidaankin asettaa säätelemään aivan toisia arvoja — vaikka kokonaan eri laitetta!

Ääntä peliin

Risto Mäki-Petäyksen suunnittelema ja toteuttama ohjelmisto moduuleineen kontrolloi MIDI-syntetisaattoreita Arduino Nano -kehitysalustan avulla. Alustaan on liitetty pyöritettävä valitsin (eng. rotary dial), joka on teknisestä näkökulmasta pulssianturi (eng. pulse encoder). Sen lisäksi siihen on kytketty LCD-näyttö ja MIDI-portit, joihin voi liittää sarjaan useampia MIDI-syntetisaattoreita. Valitsimen avulla liikutaan LCD-näytön piirtämän, kaksi tekstiriviä kattavan käyttöliittymän valikossa.

Käyttöliittymän avulla elektronisen musiikin tuottaja saa paremman otteen sellaisista laitteista, joihin ei ole valmista käyttöliittymää tai joiden nykyinen käyttöliittymä on kömpelö ja hidas. Lisäksi sen avulla pystyy kontrolloimaan useita eri laitteita samasta lähteestä. Esiteltyä ideaa voi kuitenkin soveltaa kaikenlaisiin projekteihin, joissa mikrokontrollerin toimintaa halutaan voida muuttaa lataamatta uutta koodia vanhan päälle.

Palapelin palaset

Vaikka valikossa liikutaan vain yhden pyöritettävän valitsimen avulla, käyttöliittymä on erittäin helppo- ja nopeakäyttöinen. Myötäpäivään pyörittämällä liikutaan valikon tasolla eteenpäin ja vastapäivään taaksepäin. LCD-näytön ensimmäisellä rivillä lukee nykyisen valikon tason nimi, esimerkiksi "General settings", ja toisella alavalikon nimi, esimerkiksi "MIDI channel" tai "Return". Painamalla valitsinta siirrytään alarivin osoittamaan alavalikkoon tai palataan

Mikä MIDI?

MIDI on sarjaliikenneprotokolla, jota käytetään elektronisten soitinten ohjaamiseen. Sillä voidaan lähettää nuotteja tai kontrollisignaalia. MIDI:ssä on kuusitoista kanavaa per portti, ja laitteet voidaan kytkeä sarjaan, jolloin yksi MIDI-ohjain voi ohjata kaikkia näistä laitteista. Kaikki laitteet saavat kaikki viestit, mutta ne voivat kuunnella vain yhtä kanavaa tai mielivaltaista määrää kanavia kerrallaan.

ylemmälle tasolle.

Valikon alimmalla tasolla muokataan arvoja, jolloin näytön ylin rivi näyttää arvon nimen ja alempi nykyisen arvon. Pyörittämällä voidaan muuttaa arvoa, ja painaminen palaa tasoa ylemmäs tallettaen nykyisen arvon. Yhden pyöritettävän valitsimen sijaan voisi tietenkin olla kolme painonappia, mutta pyörövalitsin mahdollistaa nopeamman liikkumisen ja arvomuutokset.

Näytön ja valitsimet saa välissä olevan moduulin ansiosta näppärästi irti ja kiinni Arduinoon. Jos kontrollerin tahtoo erilleen valikkosysteemistä syöttölaitteineen, kaikkia johtoja ei tarvitse erikseen irroittaa. Nettijatkoista löytyy moduulin rakentamisessa käytetty kytkentä- ja katkaisumaski, joka helpottaa verolevylle tehtäviä kytkentöjä.

Kurkistus konepellin alle

Koodin voi jakaa periaatteessa neljään osuuteen: valikkorakenne, näytön ohjaamiseen liittyvä kirjasto, syöttölaitteisiin liittyvät kirjastot ja MIDI-signaalin ohjaamiseen tai muuhun haluttuun toimintoon liittyvät kirjastot. Näistä valikkorakenne on projektin kannalta kiinnostavin.

Valikko-objekti, joka vastaa graafisesti yhtä näytön näkymää, on toteutettu MenuItem-tietueena (eng. struct) ja yksi valikon taso taulukkona tällaisia. MenuItemillä on kolme attribuuttia, joista kaksi ensimmäistä ovat merkkijonoja, kolmas osoitin alivalikkoon ja viimeinen on funktio-osoittimista (call back) koostuva rakenne. Merkkijonot vastaavat näytölle piirtyviä tekstejä, ja funktiot toimintoja, jotka tapahtuvat valitsinta pyöritettäessä tai painettaessa kytkimiä. Globaali pinomuuttuja pitää kirjaa siitä, missä kohtaa valikkoa liikutaan.

Valikkokoodin runko on toteutettu siten, että se skaalautuu tarpeen mukaan käyttämään useampaakin kontrollia kuin pyöritettävää valitsinta, esim. numeronäppäimistöä. MenuItem-tietueen funktio-osoitinten määrittelyyn vain pitää lisätä attribuuttina useampi palautusfunktio, jotka vastaavat uusia näppäimiä.

Valitsimeen liittyvä kirjasto tulkitsee valitsimen lähettämää kolmea signaalia ja huolehtii mm. siitä, että nopeasti myötäpäivään pyöritettäessä mikrokontrolleri ei luule valitsimen pyörivän hitaasti vastapäivään. Näin voi tapahtua, jos valitsimelta tulee pulsseja niin nopeasti, ettei kontrolleri ehdi lukea niitä kaikkia.

Käyttämämme LCD-näyttö puolestaan veisi muuten vähintään seitsemän nastaa, mutta mikrokontrollerin I/O-porttien säästämiseksi Arduino lähettää siirtorekisterille vain kolmea. Näistä yksi on datasignaali ja kaksi kellosignaaleja. Siirtorekisteri muuttaa nämä kolme signaalia näytön tarvitsemaksi seitsemäksi. Näytön ohjaamiseen liittyvä kirjasto pitää huolta siitä, että siirtorekisterille lähetetään oikean protokollan mukaista signaalia.

Entä se muokattavuus?

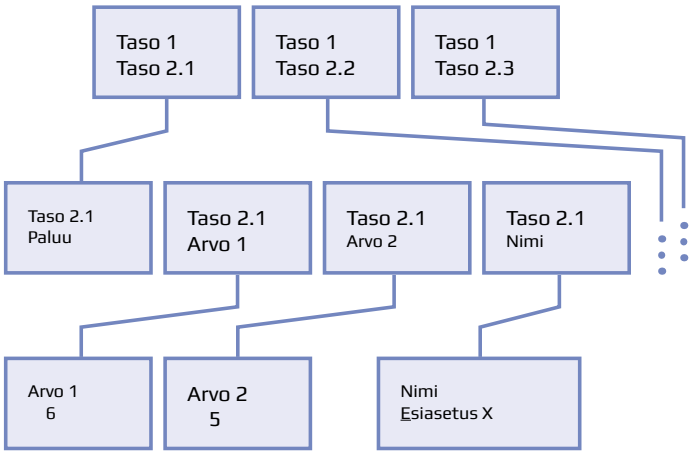
Esimerkkiprojektissa LCD-näytön reunoja kiertää kahdeksan potentiometriä, joiden avulla MIDI-laitteita ohjataan. Valikon avulla voi luoda erilaisia syntetisaattoriprofiileja. Eri profiilien ollessa valittuina potentiometrit ohjaavat eri laitetta tai eri arvoja laitteessa. Jos halutaan ohjata yhdestä laitteesta useampaa kuin kahdeksaa eri arvoa, sille on luotava useamman sivun verran profiileja.

Yhden profiilin ollessa valittuna potentiometrit saattavat siis vaikkapa ohjata syntetisaattorin oskillaattoriin liittyviä arvoja ja toisessa profiilissa samat potentiometrit sen mikseriin liittyviä arvoja. Kolmas profiili voisi laittaa potentiometrit kontrolloimaan kokonaan eri laitetta, vaikkapa sampleria.

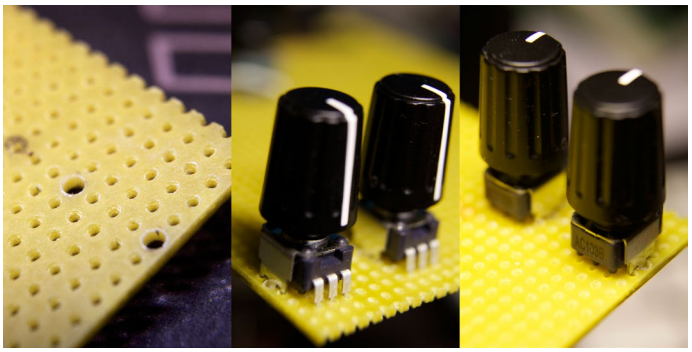
Mistä sitten voi nähdä, mitä arvoja potentiometrit milloinkin ohjaavat?

Asian selvittämiseksi ei tarvitse palata tutkimaan valikoita. Valitsimen oltua hetken koskemattomana näytölle piirtyy kuva, joka osoittaa potentiometrien nykyiset säätökohteet ja niiden arvon. Ei myöskään tarvitse opetella ulkoa, mikä esiasetus liittyy mihinkin, vaan esiasetukselle on mahdollista antaa nimi.

Nimeä muokataan valikossa siinä missä muitakin arvoja. Nimenmuokkaustilassa valitsimen pyörittäminen ohjaa tekstirivillä liikkuvaa kursoria, ja painamalla valitsinta päästään vaihtamaan valittua merkkiä. Pyörittämällä merkit



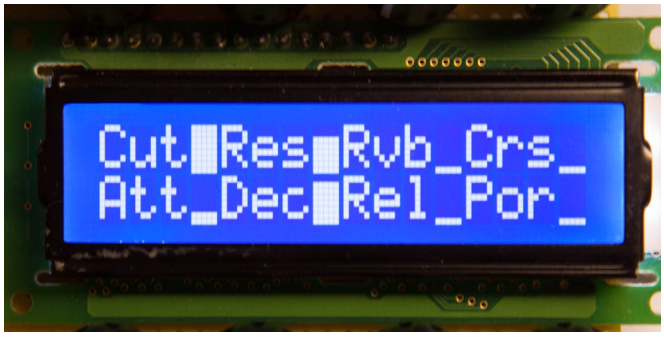
Esimerkki hierarkkisen valikon rakenteesta. Ylempi rivi ilmaisee aktiivisen valikon, alemmalla rivillä selviää kohde, johon painikkeella valitsemalla sirrytään. Syvin taso on jonkin arvon muokkaus.



Pienellä työstämisellä saa verolevylle sovitettua komponentteja, joita ei ole mitoitettu tuumarasterille.

```
struct MenuCallback {
  void (*dec_cb)(void);
  void (*inc_cb)(void);
  void (*push_cb)(void);
};

struct MenuItem {
  char *rows[2];
  struct MenuItem *subMenu;
  struct MenuCallback cb;
};

struct MenuCallback return_cb = { menudec, menuinc, menureturn
};
struct MenuCallback enter_cb  = { menudec, menuinc, menuenter
};

char m_main_title[]   = "Main Menu";
char m_preset_title[] = "Preset Edit";

MenuItem menuend = { { NULL }, NULL, {NULL, NULL, NULL} };

struct MenuItem m_main[] = {
  {{ m_main_title, m_return },       NULL,       return_cb },
  {{ m_main_title, m_preset_title }, m_reset,    enter_cb },
  {{ m_main_title, m_global_title }, m_global,   enter_cb },
  menuend
};
```

Valikon yhdestä kohdasta on tehty oma structinsa.



Näyttö oletustilassa: ruudulla näkyy parametrien nimet ja palkkien osoittamat arvot.

Esimerkki valikon kohdasta. Ylärivillä tämänhetkisen valikon nimi, alhaalla valinta.

Merkkijonon editointi, kursori näkyy muokattavan merkin kohdalla, nupin kääntö muokkaa merkkiä.

Arduino-mikrokontrollereista

Mikrokontrolleri on elektroniikan komponentti, joka sisältää samalla mikropiirillä prosessorin, ohjelmamuistia, käyttömuistia sekä syöttö- ja tulostustoimintoja. Niitä käytetään useimmiten sulautetuissa järjestelmissä, esimerkiksi kaukosäätimissä ja digitaalisissa kelloissa.

Arduino on avoimeen laitteistoon perustuva elektroniikkaalusta ja ohjelmointiympäristö. Alustan on nostanut suosioon se, että siihen on helppo liittää kolmannen osapuolen valmistamia lisäosia, kuten sensoreita, moottoreita, LED-valoja ja muita komponentteja.

Arduino-laitteet perustuvat 8-bittiseen Atmel AVR -mikrokontrolleriin, ja niitä ohjelmoidaan C++:aan perustuvalla Arduino-ohjelmointikielellä. Uusimmissa kontrollereissa ohjelmointi tehdään USB:n kautta tai Bluetoothin avulla.

Arduino-laitteita on eri malleja, joista tässä projektissa käytetty Nano on yksi pienimmistä sekä fyysisen kokonsa että esimerkiksi muistinsa puolesta.

Lähteet:
http://fi.wikipedia.org/wiki/Arduino
http://arduino.cc/en/

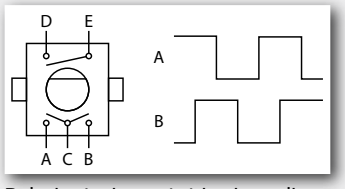
viuhuvat näytöllä järjestyksessä, ja oikean merkin löytyessä se talletetaan painamalla. Tämän jälkeen kursoria voidaan liikuttaa seuraavan muokattavan merkin kohdalle. Kun kursori viedään ulos näytöltä, nimi tallentuu ja valikossa palataan takaisin.

Tahtoo oman!

Skrollin nettijatkoista osoitteesta http://skrolli.fi/2013.3/ löydät projektista lisää materiaalia, kuten esittelyvideon, kytkentäkaavion ja koodin. Rakenteletpa vastaisuudessa sitten MIDI-kontrolleria, saunatonttua tai sokkelorobottia, muista että Arduinosta saa enemmän irti, kun sille tekee kätevän käyttöliittymän. Sen ei tarvitse olla monimutkainen – tässäkin projektissa käytettiin vain yhtä pyöritettävää namiskaa ja kaksirivistä LCD-näyttöä. Modernit tietokoneet eroavat muista elektronisista laitteista eniten siinä, että niiden toimintaan voi vaikuttaa. Laitetaan siis niistä pienimmätkin tekemään jotain monipuolista!

Pulssianturin käytöstä

Pulssiantureita tai pulssienkoodereita (eng. rotary encoder) on erilaisia, ja ne lähettävät eri protokollien mukaisia signaaleja. Tässä tapauksessa käytetään inkrementaalista, joka pyöritettäessä nuppia kulkee tiettyä sekvenssiä eteen- tai taaksepäin. Liikkeen seuranta voi olla mekaaninen tai optinen, joista mekaanisen suurin pyöritysnopeus on yleensä alhaisempi. On olemassa painonapillisia ja painonapittomia pulssiantureita. Painonapitonta voi käyttää lisäämällä ylimääräisen painokytkimen.



Pulssianturin nastat ja signaali

Oman pulssianturin protokollan voi saada selville datalehdestä (eng. data sheet), jonka voi löytää googlettamalla pulssianturin mallinumeroa, jos sellainen on tiedossa. Kaikista malleista ei kuitenkaan löydy tietoa netistä. Tällöin on otettava itse selvää protokollasta esimerkiksi yleismittarin tai oskilloskoopin avulla. Pulssienkooderin voi myös kytkeä Arduinoon ja tehdä ohjelman, joka lähettää nastojen arvot sarjaporttia pitkin koneelle. Toteuttamamme koodi tukee yleisintä nelivaiheista protokollaa sekä yhtä vastaan tullutta poikkeavaa kaksivaiheista.

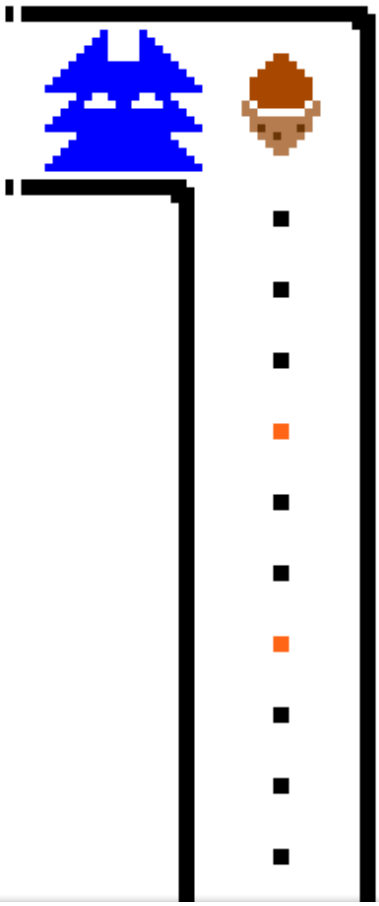
Jos pulssianturi on painonapillinen, yleensä yhden puolen nastat ilmaisevat liikkeen ja toisen puolen nastat painalluksen. Painonapin nastat voi kytkeä kummin päin tahansa: yksi nasta maahan ja toinen Arduinon sisääntulonastaan. Liikkeen ilmaisee vastapuolella kaksi kytkintä, joiden yksi yhteinen nasta on yleensä keskimmäinen, ja vastakkaiset nastat ovat reunoilla.

Projektimme koodissa on kytketty päälle Arduinon sisäinen ylösvetovastus, jolloin keskimmäisen nastan sekä painonapin toisen nastan voi kytkeä maahan. Muussa tapauksessa tarvitsisi käyttää erillistä ylös- tai alasvetovastusta. Käsittelemme aihetta syvällisemmin nettijatkoissa.

Lisätietoa pulssiantureista yleensä esimerkiksi:
http://en.wikipedia.org/wiki/Rotary_encoder#Incremental_rotary_encoder

Pähkinänurkka

Tällä palstalla ilmestyy ohjelmointiin ja matematiikkaan liittyviä tehtäviä. Kaikki tehtävät on mahdollista ratkaista nopeasti sopivalla algoritmilla — ja osa ilman tietokonetta!

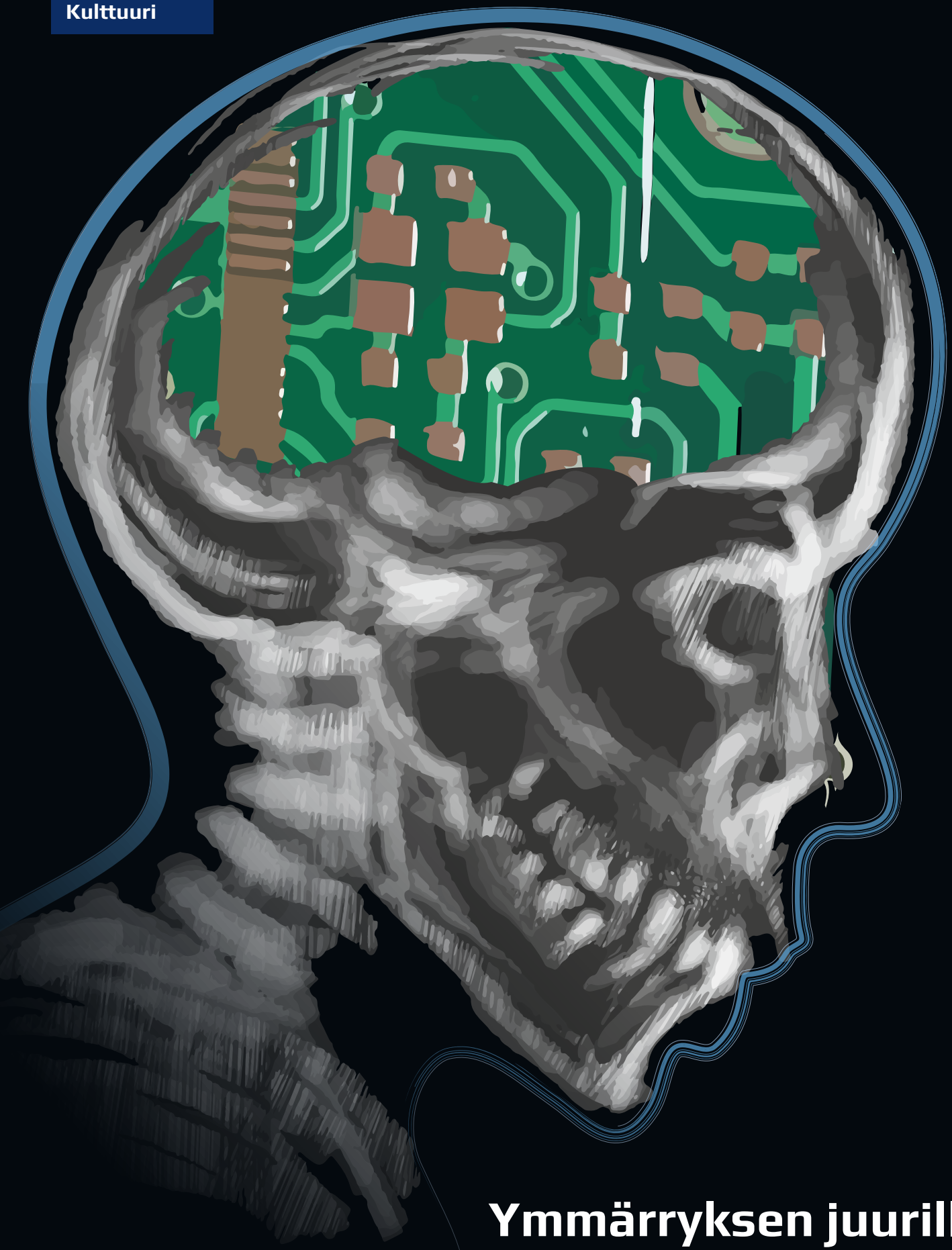


Uolevi on koodannut ohjelman, jolla voi lähettää viestejä netin välityksellä. Jokainen lähetettävä viesti on bittijono eli muodostuu merkeistä 0 ja 1. Ikävä kyllä Uolevin ohjelmassa on bugi ja osa biteistä saattaa kadota matkan varrella. Kuitenkin kaikki perille tulleet bitit ovat samassa järjestyksessä kuin alkuperäisessä viestissä ja ainakin yksi bitti tulee perille.

Esimerkiksi jos Uolevi lähettää viestin 1101, vastaanottaja voi saada 9 erilaista viestiä: 0, 1, 01, 10, 11, 101, 110, 111 tai 1101. Uolevi haluaa tutkia ohjelmansa luotettavuutta laskemalla, montako erilaista viestiä voi syntyä eri tilanteissa. Voitko auttaa Uolevia?

1. Jos Uolevi lähettää viestin 0110001111, montako erilaista viestiä vastaanottaja voi saada?
2. Entä jos Uolevi lähettää viestin 01100 01111000001111110000000111111 1000000000?
3. Vastaanottaja voi saada tarkalleen 123 erilaista viestiä. Kuinka pitkä on lyhin mahdollinen Uolevin lähettämä viesti?
4. Entä jos vastaanottaja voi saada tarkalleen 999999 erilaista viestiä?
5. Uolevi lähettää viestin, jossa on n bittiä. Mikä on odotusarvo sille, kuinka monta erilaista viestiä vastaanottaja voi saada, jos kaikki viestit ovat yhtä todennäköisiä?

Ongelmien ratkaisut löytyvät Skrollin webisivuilta osoitteesta:
http://skrolli.fi/pahkina/201303/



Ymmärryksen juurilla

eli kuinka todellisuus selitetään koneelle

Voisivatko tietokoneet ymmärtää maailmaa? Mitä ymmärtäminen ylipäänsä tarkoittaa? Koneymmärtämisen pohjatyö on tehty jo 60–70-luvulla.

Teksti: Panu Kalliokoski Kuvat: Mitol Berschewsky

Voisivatko tietokoneet ymmärtää maailmaa? Mitä ymmärtäminen ylipäänsä tarkoittaa? Koneymmärtämisen pohjatyö on tehty jo 60–70-luvulla. Tässä artikkelissa tutustumme klassisiin merkitysteorioihin ja tietämyksen esittämismuotoihin.

Käsitys siitä, mitä tietokoneelta voi odottaa, on muuttunut ajan kuluessa. Enimmäkseen vaatimukset ovat kasvaneet: tietokoneilta odotetaan aina vain suurempaa laskentatehoa, tallennus- ja tiedonvälityskykyä. On kuitenkin asioita, joissa ihmisten odotukset tietokoneiden suhteen ovat laskeneet. Lähes kukaan ei enää odota tietokoneiden ymmärtävän ihmisten kieltä, tunnistavan näkyykö kuvassa vettä tai arvioivan eri lähteiden luotettavuutta. Nykyajan tietokoneilla lasketaan kymmeniä kolmiulotteisen maailman projektioita sekunnissa, mutta 70–90-luvulla vaativimmat ohjelmat olivat ne, jotka yrittivät ymmärtää monimutkaista maailmaa. Tässä artikkelissa tarkastellaan yleisellä tasolla näiden ohjelmien toimintaa.

Asioiden ymmärtäminen on monimutkainen, mutta ei sentään mahdoton ongelma. Ymmärtävät ohjelmat ovat olleet niin pitkään unohduksissa, että moni arvioi tällaisen ohjelman kehittämisen vaikeammaksi kuin se onkaan. On totta, että tekoälyn kultavuosina oltiin puolestaan aiheettoman optimistisia. Tuohon aikaan aliarvioitiin reippaasti, miten vaikeaa on tehdä ohjelmia, jotka oppivat, osaavat yhdistää kokemuksen käsitteisiin ja selviävät luonnollisen kielen epätarkkuuksien ja monitulkintaisuuden kanssa. Silti jo 80-luvulla pystyttiin tekemään luonnollisella kielellä ohjattuja ohjelmia, jotka:

- tuottavat automaattisesti virheettömiä käännöksiä tiettyjen aihepiirien lehtiartikkeleista
- osaavat keksiä syitä ja välivaiheita kertomuksessa tapahtuville asioille (*SAM* ja *PAM*)
- osaavat perustella toimintatapaansa virtuaalimaailmassa (*SHRDLU*)
- tekevät yleistyksiä havaintojen perusteella
- tuottavat tarkempia diagnooseja epävarman tiedon perusteella kuin ihmislääkärit (*Mycin*).

Miten nämä ohjelmat oikein toimivat? Mitä tarkoittaa, että ne "ymmärsivät" jotain? Nämä luonnollista kieltä käsittelevät ohjelmat toimivat tyypillisesti siten, että ne jäsensivät luonnollisen kielen ilmauksia joksikin sisäiseksi tietorakenteeksi, joka edustaa ilmausten tietosisältöä. Tällaista tietorakennetta kutsutaan tietämyksen esitysmuodoksi (knowledge representation). Näille tietämyksen esitysmuodoille tehtiin sitten erilaisia operaatioita, kuten etsittiin tietokannoista niihin liittyviä lisätietoja tai päätelmiä, muunnettiin niitä takaisin (jollekin toiselle) luonnolliselle kielelle, lisättiin niiden tiedot tietämystietokantaan tai suunniteltiin tulevaa toimintaa tavoitteiden perusteella.

Tietämyksen esitysmuodot eivät ole mielenkiintoisia ainoastaan ohjelmoinnin kannalta. Tähän päivään asti nämä tekoälyohjelmien käyttämät sisäiset esitysmuodot ovat huolellisimpia ja käytännönläheisimpiä teorioita siitä, mitä merkitys oikeastaan on. Ne ovat myös ainoita teorioita ihmisen toiminnasta, joita on onnistuttu soveltamaan ihmisen kaltaisen toiminnan tuottamiseksi. Nykyajan psykologia ja kognitiotiede ovat hyötyneet myös tekoälytutkimuksen epäonnistumisesta, sillä tietämyksen esitysmuotojen puutteet ja ongelmat ovat tuoneet esiin, miten ihmisen mieli **ei** toimi. Seuraavaksi tarkastelemme muutamia keskeisimpiä perinteisiä tietämyksen esitysmuotoja.

Predikaattilogiikka (PL)

Logiikkaa tutkittiin "ajattelun kielenä" jo kauan ennen tietokoneita. 60-luvulla John Robinson ja muut kehittivät muutamia olennaisia työvälineitä, joiden avulla rakennettiin ensimmäiset päättelyalgoritmit: logiikkalauseiden konjunktiivinen normaalimuoto (CNF) ja sille sovellettava yleispätevä päättelysääntö, resoluutio. Tämä merkitsi, että koneet pystyivät nyt tuottamaan automaattisesti väitejoukon kaikki seuraukset. Tarvitsisi vain kirjoittaa yleisluontoiset säännöt sekä havainnot logiikkalausekkeiksi.

Olettakaamme esimerkiksi, että ohjelma tuntee säännöt:

1. ne, jotka oleskelevat ravintolassa, mutta eivät ole siellä töissä, ovat ravintolan asiakkaita
2. ravintolan asiakas maksaa ravintolan työntekijälle

Predikaattilogiikan lausekkeina nämä voitaisiin ilmaista muodossa:

```
(seuraa (asiakas ?x ?y)
      jos (ihminen ?x)
        (ravintola ?y)
        (sijaitsee ?x ?y)
        (ei (töissä ?x ?y)))
(seuraa (maksaa ?x ?y)
      jos (asiakas ?x ?z)
        (ravintola ?z)
        (töissä ?y ?z))
```

Nyt jos ohjelmalle syötetään tieto "Tepa on ihminen, sijaitsee Krouvi-ravintolassa eikä ole siellä töissä", kone osaa päätellä "Tepa maksaa kaikille, jotka ovat töissä Krouvi-ravintolassa" eli

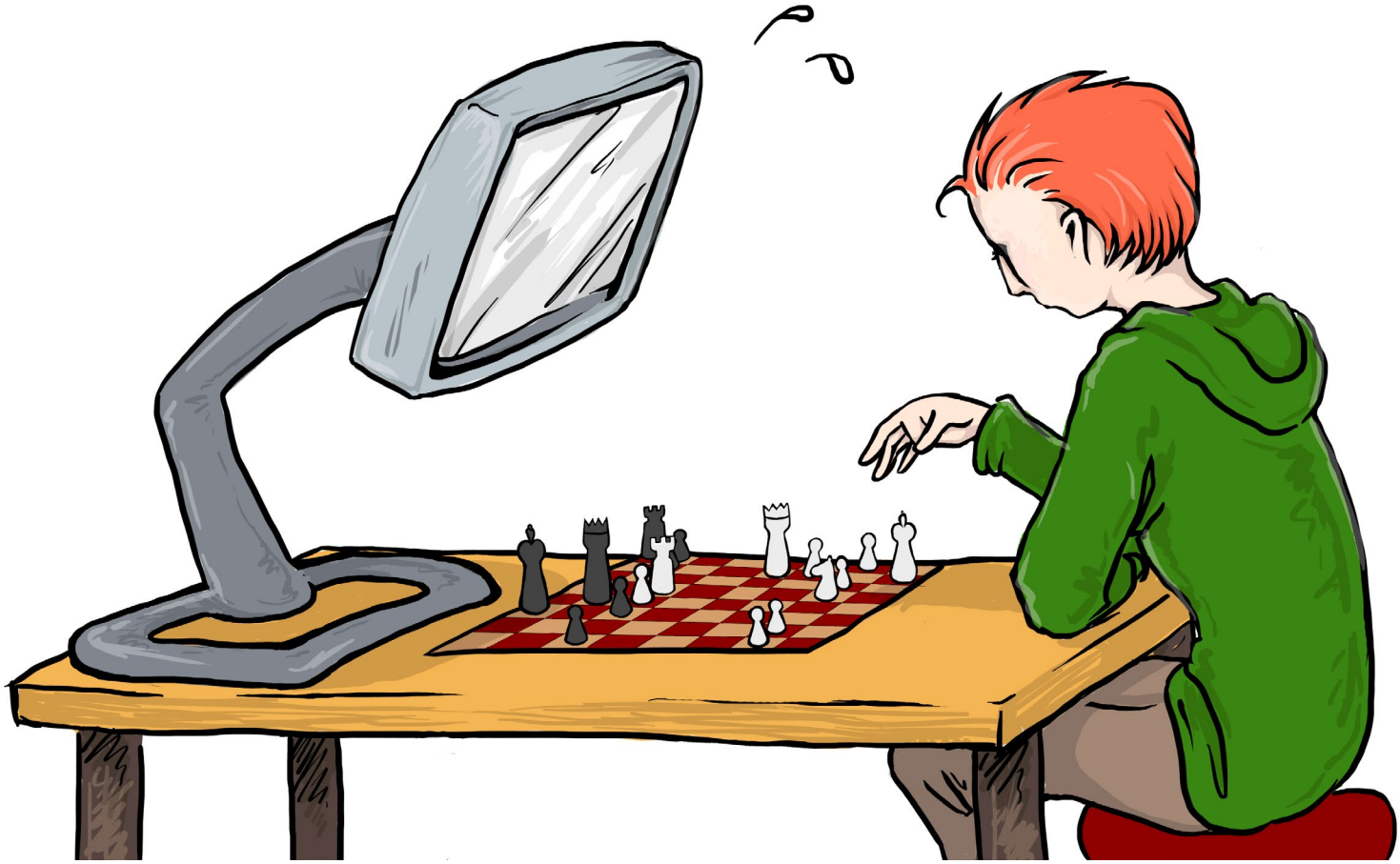
```
(seuraa (maksaa Tepa ?y)
      jos (töissä ?y Krouvi))
```

Koska logiikka tarjoaa tällaisen tarkan tavan ilmaista sääntöjä ja soveltaa niitä, tämä oli yksi ilmeinen lähestymistapa luonnollisen kielen käsittelyyn: muunnetaan luonnollisen kielen ilmauksia logiikkalausekkeiksi, tehdään niistä päätelmät, ja muunnetaan uudet logiikkalausekkeet takaisin luonnollisen kielen ilmauksiksi. Osoittautui kuitenkin, että logiikka oli hankala väline sääntöjen (ja joskus havaintojenkin) ilmaisemiseen.

Ensinnäkään ihmisten käyttämät päättelysäännöt eivät usein ole täydellisen varmoja, vaan ne perustuvat erilaisiin pohjaoletuksiin. Yllä olevissa säännöissä on oletettu, että kaikki ravintolat toimivat samaan tapaan, asiakkaita on vain yhdenlaisia, eikä kukaan oleskele ravintolassa muista syistä. Mitkä näistä oletuksista ovat todella varmoja, riippuu käsitteiden "ravintola", "asiakas" ja muiden määritelmistä. Nämä epävarmuudet voidaan ilmaista logiikassa, jolloin yllä olevaan johtopäätökseen lisättäisiin varaukset "... tai Tepa on epänormaali asiakas tai Krouvi on epänormaali ravintola", mahdollisesti vieläkin useampia varauksia. Muita ratkaisuja epävarmuuteen ovat muun muassa oletuslogiikat ja todennäköisyyslogiikat.

Toisekseen sanoja on usein vaikea muuntaa logiikan käsitteiksi. Tämä johtuu siitä, että sanat viittaavat usein epämääräisesti monenlaisiin asioihin eikä ole helppoa päätellä, mikä niiden tarkka merkitys on. Logiikkalausekkeiden on kuitenkin oltava ehdottoman paikkansapitäviä, tai muuten niistä seuraa helposti virhepäätelmiä. Luonnollisen kielen lauseita on sitä vaikeampi muuttaa predikaattilogiikaksi, mitä tarkemmin määriteltyjä predikaattilogiikassa käytetyt suhteet ovat. Mutta mitä huonommin nämä suhteet on määritelty, sitä vaikeampaa on kirjoittaa virheettömiä päättelysääntöjä.

Kolmanneksi predikaattilogiikka soveltuu melko hyvin ikuisten, muuttumattomien asioiden kuvaamiseen, mutta teot, tapahtumat ja aikakehitykset saavat logiikassa hyvin monimutkaisen muodon. Esimerkiksi yllä olevat säännöt tarvitsisivat tarkenteita siihen, että Tepa on asiakas nimenomaan siihen aikaan, **jolloin** sijaitsee ravintolassa. Koska suuri osa luonnollisesta



kielestä on erilaisten tilanteiden ja niiden muutosten kuvaamista, monet tekoälytutkijat kehittivät mieluummin omia, käytännönläheisempiä tietämyksen esitysmuotoja.

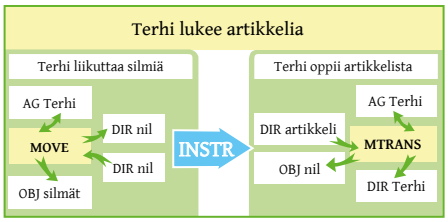
Logiikka kuitenkin teki mahdolliseksi joitakin pysyviä saavutuksia tekoälyn saralla. Yksi näistä oli se, että erilaiset logiikkaa käyttävät tekoälysovellukset voitiin jakaa kolmeksi osa-alueeksi: tietämyskannaksi, päättelymoottoriksi ja näitä hyödyntäväksi käyttöliittymäksi. Näin pystyttiin tuottamaan uudelleenkäytettävämpiä komponentteja, jotka auttoivat muun muassa monien asiantuntijajärjestelmien kehittämisessä.

Conceptual Dependency (CD)

Conceptual dependency -esitysmuodon kehitti Roger Schank yhdessä muiden kanssa 60--70-luvun vaihteessa kuvaamaan erilaisten kertomusten asiasisältöä ja auttamaan päätelmien tekemisessä asiasisällöstä. Sen lähtökohtana oli luoda kohtalaisen yleispätevä esitysmuoto, jossa muutamista yksinkertaisista palikoista voisi yhdistelemällä tuottaa kaikki erilaiset merkitykset.

Alkuperäisessä CD-esitysmuodossa primitiivilauseessa oli aina teko, tekijä, teon kohde, lähtöpaikka ja kohdepaikka. Erilaisia tekoja oli vain 11 erilaista, sellaisia kuin siirtyminen (PTRANS), fyysisen voiman kohdistaminen (PROPEL), ajatuksen keksiminen (MBUILD) ja yhdistyminen johonkin (INGEST). Monimutkaisemmat ilmaukset muodostettiin tällaisista primitiivilauseista yhdistelemällä niitä lauseiden välisillä suhteilla: yksi teko saattoi olla toisen kohde, syy tai väline.

Esimerkiksi alla on CD-esitykset ilmauksista "Terhi liikuttaa silmiä", "Terhi oppii artikkelista" ja "Terhi lukee artikkelia":




```
(MOVE Terhi silmät nil nil)
(MTRANS Terhi nil artikkeli Terhi)
(INSTR (MOVE Terhi silmät nil nil)
(MTRANS Terhi nil artikkeli Terhi))
```

Termien selitykset:
MOVE – tekijä liikuttaa jotain osaansa;
MTRANS – tekijä siirtää ajatuksia tai ideoita;
AG – tapahtuman aiheuttaja, tekijä
OBJ – se, johon tapahtuma vaikuttaa, kohde;
DIR – suuntamääreet, mistä mihin jotain tapahtuu;
nil – määrittelemätön lauseenjäsen
INSTR – instrumenttisuhde, yksi teko on toisen väline)

Alun perin CD-muotoa käytettiin välimuotona käännettäessä tekstiä kielten välillä. Myöhemmin siihen lisättiin päättelymekanismeja, jotka tuottivat erilaisista tilanteista arvauksia ja vaihtoehtoisia ilmauksia esimerkiksi sen perusteella, millaisia tyypillisiä seurauksia niillä on tai minkälaisessa asiainkulussa eli skriptissä ne olisivat tyypillisiä tilanteita.

CD-muodolla saatiin testatuksi monia luonnollisen kielen käsittelyyn liittyviä teorioita ja ideoita. CD-pohjaiset järjestelmät eivät kuitenkaan tuottaneet olennaisia kaupallisia sovelluksia. Yksi syy lienee se, että alkuperäinen oletus 11 perustekotyypistä ei ollut kestävä. Varsinkin ihmisten sosiaalisissa suhteissa on monia tekoja, kuten vaikkapa ehdottaminen, painostaminen ja isyyden tunnustaminen, joiden esittäminen CD-lausekkeina on hyvin monimutkaista. CD-esitysmuotoa laajennettiin jonkin aikaa uusilla käsitteillä, mutta laajempi primitiivijoukko teki CD-rakenteiden käsittelystä hankalampaa.

CD:tä ei ollut koskaan suunniteltu esittämään maailmassa vallitsevia tiloja, ainoastaan tapahtumia. Kun maailman tilan kuvaamiseen tarkoitetut esitysmuodot alkoivat monimutkaistua, monet alkoivat käyttää niitä myös tapahtumien esittämiseen.

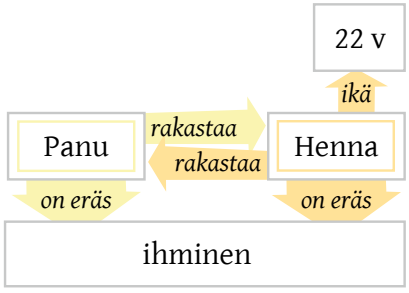
Semanttiset ja kehysverkot

Rakkaalla lapsella on monta nimeä. Samaa tai lähes samaa asiaa on aikojen saa-

tossa kutsuttu ainakin semanttisiksi verkoiksi, kehysverkoiksi, konseptuaalisiksi graafeiksi ja suhdekaavioiksi. Semanttiset verkot ovat tietämyksen esitysmuoto, jossa luetellaan erilaisia käsitteitä, niiden välisiä suhteita ja ominaisuuksia.

Semanttinen verkko voi olla vain yksi esitystapa predikaattilogiikan lausekkeille. Sellaisissa kaikki suhteet ovat binäärisiä eli kahdenvälisiä. Monet tekoälyissä käytetyt semanttiset verkot sisältävät kuitenkin muitakin mahdollisuuksia, kuten ominaisuuksien oletusarvoja, tarvittaessa dynaamisesti laskettuja ominaisuuksia tai muuta vastaavaa.

Tässä on esimerkki semanttisesta verkosta. Verkko edustaa väitteitä "Panu ja Henna ovat ihmisiä", "Panu ja Henna rakastavat toisiaan" ja "Henna on 22 vuotta vanha".



Termien selitykset:
on eräs – tarkoittaa, että yksilö X kuuluu asialuokkaan Y)

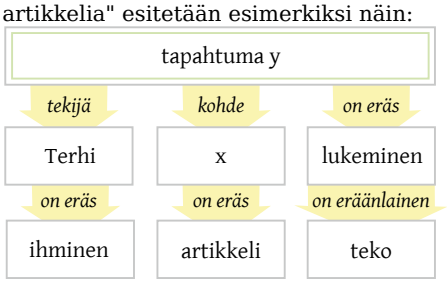
Semanttisten verkkojen ja kehysverkkojen teoria palautuu 1900-luvun alkupuolelle ja siis selkeästi aikaan ennen tietokoneita. 70-luvun puolivälissä Marvin Minsky ja John Sowa kehittivät näihin vanhoihin ajatuksiin perustuvia esitysmuotoja tekoälyohjelmia varten. Tavoitteena oli myös jäljitellä sitä, miten ihminen järjestelee mielessään tietoja suuriksi, yhteen kuuluviksi kokonaisuuksiksi. Yhtä käsitettä suhteineen ja ominaisuuksineen kutsuttiin "kehykseksi", mistä nimi "kehysverkko" tulee. Kehysteoria oli sittemmin yhtenä esikuvana oliopohjaiselle ohjelmointiparadigmalle samoin kuin relaatiotietokantojen tietomalleille. Nykymaailmassa UML-diagrammit tarjoavat vakiintuneen esitystavan semanttisille verkoille.

Semanttisissa verkoissa käsitteet olivat usein prototyyppisiä. Se tarkoittaa, että esimerkiksi käsite "ihminen" sisälsi tietoja tyypillisestä ihmisestä sen sijaan, että edustaisi vain kaikkien ihmisten luokkaa. Päättely semanttisessa verkossa on usein vastauksen etsimistä johonkin kysymykseen linkkejä seuraamalla, jolloin ensimmäinen vastaan tullut vastaus yliajaa muut mahdolliset vastaukset. Esimerkkejä tällaisista päätelmistä voisivat olla:

1. Jos selvitetään, montako jalkaa Panulla on, vastaus ei löydy Panusta itsestään, mutta seuraamalla "on eräs" -linkkiä ihmisen käsitteeseen saadaan tieto, että ihmisillä on yleensä 2 jalkaa. Koska tieto löytyy täältä, ei tarvitse seurata "on eräänlainen" -linkkiä ihmisen käsitteestä nisäkkään käsitteeseen, jossa olisi taas tieto, että nisäkkäillä on yleensä 4 jalkaa.
2. Jos selvitetään, mitä tekemistä Panulla on Helsingin kanssa, löytyisi ehkä ensimmäiseksi tieto, että Panun asuinpaikka on Helsingissä, sillä näiden kahden välinen polku on vain kaksi askelta pitkä: Panu asuu talossa, joka sijaitsee Helsingissä. Paljon muitakin yhteyksiä voi olla, mutta tämä on olennaisin assosiaation lyhyyden perusteella.

Semanttiset verkot eivät sovellu erityisen hyvin päättelysääntöjen kuvaamiseen. Niihin on yleensä lisätty erityisominaisuuksia erilaisten päättelysääntöjen toteuttamisen tueksi, kuten dynaamiset ominaisuudet määrittämään, miten yksi ominaisuus (kuten ikä) riippuu toisesta (kuten syntymäajasta). On kuitenkin monenlaista yleistietoa, joka täytyy esittää muin keinoin. Semanttiset verkot ovat silti erittäin monipuolinen tietämyksen esitysmuoto. Yksi moderni semanttisten verkkojen järjestelmä on Hermann Helbigin MultiNet.

Myös tapahtumien kuvaamiseen semanttiset verkot soveltuvat. "Terhi lukee artikkelia" esitetään esimerkiksi näin:



Termien selitykset:
tekijä – tapahtuman aiheuttaja;
kohde – se, johon tapahtuma vaikuttaa;
on eräänlainen – tarkoittaa, että asialuokka X on asialuokan Y aliluokka, eli kaikki X:n yksilöt ovat myös Y:n yksilöitä

Resource Description Framework (RDF)

RDF on melko uusi, World Wide Web Consortiumin (W3C) kehittämä konekäsiteltävän tiedon esitysmuoto, joka on erityisesti suunniteltu auttamaan tiedon tuottamista hajautetusti yhteistyössä. Pohjimmiltaan se on ominaisuuksiltaan hyvin yksinkertainen semanttinen verkko. RDF-verkot koostuvat kokonaan kolmikoista (kaksi käsitettä ja niiden välinen suhde), eivätkä ne sisällä mitään perinteisiä helpotuksia päätelmien tai sääntöjen esittämiseen. RDF-verkoissa pystyy tosin viittaamaan verkon kaariin käsitteinä, mutta tätä ominaisuutta (*reifiointia*) ei hyödynnä juuri kukaan. Käytännössä se tekee mahdolliseksi väitteet, jotka kommentoivat toisia väitteitä, kuten "on hyvä, että Terhi lukee artikkelia".

RDF:ssä olennaisin uudistus vanhoihin tietämyksen esitystapoihin verrattuna on se, että sekä käsitteillä että niiden välisillä suhteilla on maailmanlaajuisesti yksiselitteiset URI-tunnisteet. Näiden tunnisteiden perusteella pystyy helposti viittaamaan muiden määrittämiin käsitteisiin ja suhteisiin ja siten laajentamaan yhtenäistä, maailmanlaajuista tietämysverkkoa. Lisäksi näitä tunnisteita voi käyttää myös linkkeinä, joista voi hakea lisätietoa käsitteestä kone- tai ihmisluettavassa muodossa.

Käyttötarkoitukseksi esitysmuodolle ei niinkään ole ajateltu tilannetta, jossa luonnollisesta kielestä tuotettaisiin automaattisesti RDF-verkkoja. Pikemminkin on ajateltu tapauksia, joissa jo olemassa olevaa koneluettavaa tietoa julkastaisiin RDF-muotoisena, jolloin sitä pystyisi yhdistelemään muihin RDF-muotoisiin tietoihin. Aika näyttää, kuinka paljon tällaista tietoa saadaan tarjolle.

Ei tässä vielä kaikki

Arkijärkeä soveltavien, arvauksia tekevien tai luonnollista kieltä ymmärtävien ohjelmien kirjoittaminen ei ole mitään mustaa magiaa. Todellisen maailman ymmärtäminen on paljon tutkittu ongelma, jossa on saatu paljon hyödyllisiä tuloksiakin.

Varsinkin deklaratiivisen eli kuvailevan tietämyksen esitysmuodoiksi on tarjolla jo varsin hyviä ratkaisuja. Tässä artikkelissa on esitelty vain näitä. Proseduraalinen eli toiminnallinen tietämys on sen sijaan vähemmän tutkittu alue, eikä tässä jutussa paneuduttu siihen.

Suomen joukkue vuoden 2013 ohjelmointiolympialaisissa Australiassa. Vasemmalta oikealle Henrik Lievonen, Kalle Luopajärvi ja Sami Kalliomäki.

Parhaasta koodista kultaa

Kisakoodaus on koodarien oma urheilulaji, jossa tehokkaat algoritmit ovat kaikki kaikessa ja bugisen koodin tekijän kohtalo on kova.

Teksti: Antti Laaksonen

Koulujen ja yliopistojen koodauskisoilla on pitkät perinteet. Koululaisten ohjelmointiolympialaiset on järjestetty vuodesta 1989 asti, ja niihin osallistuu nykyään noin 80 maata. Jotkin yritykset käyttävät kisoja rekrytointikanavana. Esimerkiksi Googlen Code Jamiin osallistuvalle voi tulla kisan jälkeen työtarjous Googlelta. Viime vuosina nettiin on ilmestynyt lukuisia kaikille avoimia kisasivustoja. Vaikuttaakin siltä, että kisakoodauksen suosio on kasvussa.

Koodauskisoissa suunnitellaan ja toteutetaan tehokkaita algoritmeja annettuihin ongelmiin. Kisoissa menestyminen vaatii sekä teorian että käytännön taitoja. Vaikeinta on keksiä, kuinka ratkaista annettu ongelma tehokkaasti. Usein algoritmin täytyy käsitellä suuria syötteitä sekunneissa. Arvostelu tapahtuu toteutetun algoritmin perusteella, joten myös itse ohjelmointivaihe on tärkeä. Jos koodi puuttuu tai ei toimi, ratkaisulla ei ole kisassa arvoa.

Voimalla tehokkaan algoritmin

Tällä hetkellä netin aktiivisin kisakoodaussivusto on venäläinen Codeforces, joka aloitti toimintansa muutama vuosi sitten. Sivustolla järjestetään noin kerran viikossa kaikille avoimia ohjelmointikisoja. Yleensä kisassa on kaksi tuntia aikaa ratkaista viisi algoritmitehtävää. Jokaiseen tehtävään tulee lähettää algoritmin toteuttava koodi, joka lukee lähtötiedot ja tulostaa vastauksen tehtävänannon mukaisessa muodossa.

Codeforcesin arvostelu on ankara. Koodista saa pisteitä vain, jos se on toimiva. Jos koodissa on pienikin bugi, tulos on yleensä nolla pistettä. Kun koodin lähettää kisan aikana palvelimelle, sen toimintaa testataan muutamalla syötteellä. Jos tässä vaiheessa löytyy bugi, voi uuden koodin vielä lähettää korvaamaan edellistä. Kisan jälkeen seuraa varsinainen arvostelu, jossa koodia koetellaan suurella määrällä syötteitä. Koodi hyväksytään, jos se toimii oikein kaikissa testeissä.

Koodaamisen lisäksi kisan aikana voi hakkeroida muiden koodeja. Jos uskoo oman ratkaisun tehtävään olevan oikea, voi ratkaisun lukita, minkä jälkeen sitä ei voi enää muuttaa. Tällöin pääsee tutkimaan muiden lähetyksiä kyseiseen tehtävään. Jos toisen koodista löytyy bugi, sen voi paljastaa antamalla syötteen, jossa koodi toimii väärin. Hakkeroinnista on hyötyä molemmille osapuolille. Bugin löytäjä palkitaan bonuspistein ja virheellisen koodin tekijä voi yrittää korjata ratkaisuaan.

Koodarin taitoa mittaa rating, joka muistuttaa shakissa käytettyä vahvuuslukua. Aluksi rating on 1500, ja muuttuu jokaisen kisan jälkeen. Hyvin mennyt kisa nostaa ratingia ja huonosti mennyt laskee. Jos rating on 1700 tai enemmän, kuuluu kilpailija eliittiin ja saa osallistua 1. divisioonan kisoihin. Näissä kisoissa kaikki tehtävät ovat vaikeita. Usein yksikään osallistuja ei onnistu kisan aikana ratkaisemaan kaikkia tehtäviä.

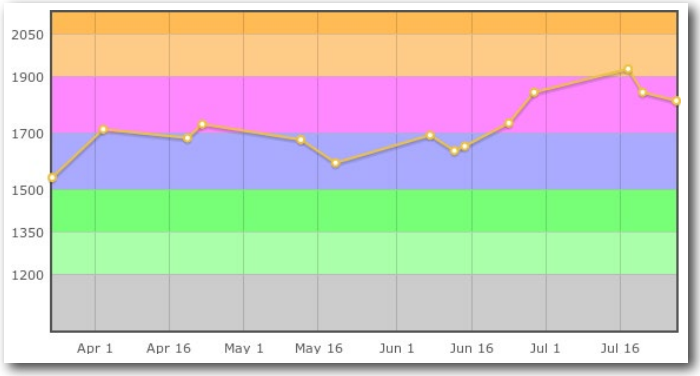
Millaisia tehtävät ovat?

Eräässä Codeforcesin kisassa helpoin tehtävä oli seuraava:

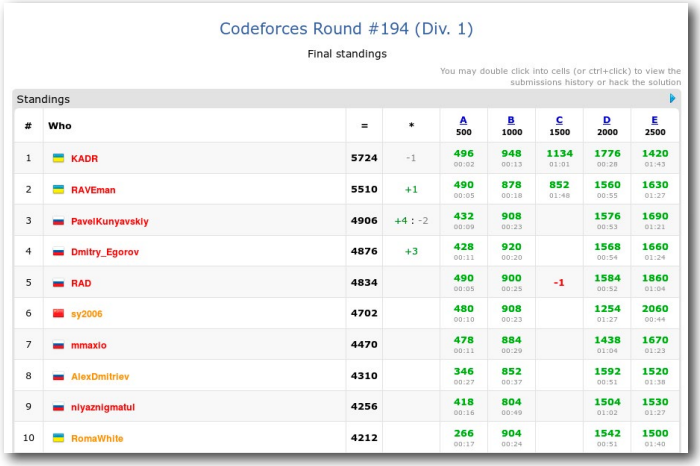
Luku on taikaluku, jos sen voi muodostaa liittämällä peräkkäin lukuja 1, 14 ja 144. Esimerkiksi 14144, 141414 ja 1411 ovat taikalukuja, kun taas 1444, 514 ja 414 eivät ole. Syötteenä on kokonaisluku n, joka on välillä 1...109. Jos luku on taikaluku, tulosta YES, ja muuten NO.

Tehtävän ratkaisuun on monia erilaisia tapoja. Tässä on yksi tiivis C++-ratkaisu, jossa on ideana poistaa luvun lopusta 1, 14 tai 144 niin kauan, kuin tämä on mahdollista.

```
#include <iostream>
using namespace std;
int n;
int main() {
    cin >> n;
    while (true) {
        if (n%10 == 1) n /= 10;
        else if (n%100 == 14) n /= 100;
        else if (n%1000 == 144) n /= 1000;
        else break;
    }
    if (n == 0) cout << "YES\n";
    else cout << "NO\n";
}
```



Codeforces-koodarin kehitystä voi tarkastella profiilissa olevasta käyrästä. Käyrän pisteitä ovat koodarin rating-arvot kisoissa. Kuvan koodari on onnistunut nousemaan rajan 1700 yli divisioonaan 1.



Codeforces Round #194 (Div. 1)

Final standings

You may double click into cells (or ctrl+click) to view the submissions history or hack the solution

Standings

#	Who	=	*	A 500	B 1000	C 1500	D 2000	E 2500
1	KADR	5724	-1	496 00:02	948 00:13	1134 01:01	1776 00:28	1420 01:43
2	RAVEman	5510	+1	490 00:05	878 00:18	852 01:48	1560 00:59	1630 01:27
3	PavelKunyavskiy	4906	+4 : -2	432 00:09	908 00:23		1576 00:53	1690 01:21
4	Dmitry_Egorov	4876	+3	428 00:11	920 00:20		1568 00:54	1660 01:24
5	RAD	4834		490 00:05	900 00:25	-1	1584 00:52	1860 01:04
6	sy2006	4702		480 00:10	908 00:23		1254 01:27	2060 00:44
7	mmaxio	4470		478 00:11	884 00:20		1438 01:04	1670 01:23
8	AlexDmitriev	4310		346 00:27	852 00:37		1592 00:51	1520 01:38
9	niyaznigmatul	4256		418 00:16	804 00:49		1504 01:02	1530 01:27
10	RomaWhite	4212		266 00:17	904 00:24		1542 00:51	1500 01:40

Codeforces-kisa on päättynyt ja tulokset on julkaistu. Ensimmäisenä oleva koodari ratkaisi tehtävän A vain kaksi minuuttia kisan alkamisen jälkeen. Tuloslistan kautta voi myös tutustua muiden lähettämiin koodeihin.

Länsimaiden perikato

Codeforcesin korkeimman ratingin haltija on valkovenäläinen Gennady Korotkevich. Hän on kisakoodausmaailman julkkis, jonka saavutuksiin kuuluu kuusi kultamitalia koululaisten ohjelmointiolympialaisista. Valko-Venäjän lisäksi Codeforcesin kärjessä ovat Venäjä, Kiina, Japani, Taiwan, Puola ja Ukraina. Suomalaisia ei huipulla näy, eikä muitakaan länsimaita. Koodauskisojen ylivoimaiset hallitsijat tulevat Itä-Euroopasta ja Aasiasta.

Yksi mahdollinen syy länsimaiden heikkoon menestykseen on, etteivät ohjelmointi ja algoritmiikka kuulu koulujen opetusohjelmaan. Suomessa yliopiston syventävillä kursseilla saatetaan opettaa samoja asioita kuin Itä-Euroopassa ohjelmointipainotteisissa kouluissa. Toisaalta koodauskisoissa menestyminen vaatii paljon työtä ja vahvan motivaation. Algoritmien harjoittelun motivaationa voi toimia halu päästä ulkomaille hyvään yliopistoon tai työpaikkaan.

Skrollin lukijoiden joukossa on epäilemättä monia potentiaalisia kisakoodareita. Tällä hetkellä Codeforcesin käyttäjissä on vain kymmenkunta suomalaista, mutta nyt on mahdollisuus muuttaa tilanne ja koodata Suomi maailmankartalle.

Haluatko olympialaisiin?

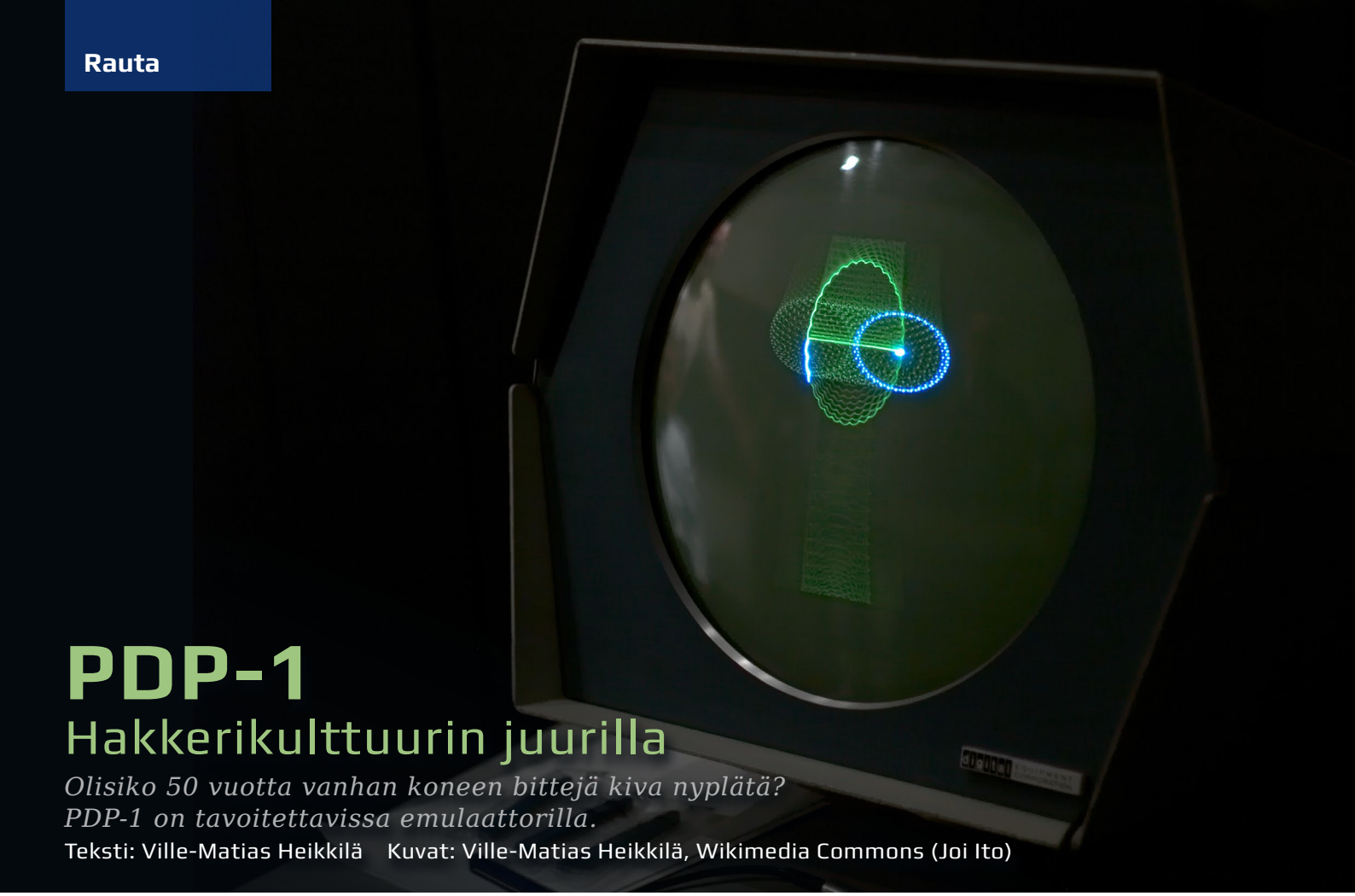
Suomi osallistuu vuosittain koululaisten ohjelmointiolympialaisiin (IOI). Kukin maa voi lähettää olympialaisiin neljän hengen joukkueen. Jos olet peruskoulussa tai lukiossa, sinulla on mahdollisuus päästä Suomen joukkueeseen. Ensimmäinen vaihe on osallistua Datatähti-kisaan, joka alkaa tänä vuonna 15. lokakuuta. Sen perusteella valitaan osallistujat valmennukseen ja aikanaan Suomen joukkue olympialaisiin Taiwaniin heinäkuussa 2014.

Kuka näin ässää lehteä tekee?!

Jatkossa vaikka sinä! Skrollin alkutaipaleen legendaaristen avustajien joukkoon mahtuu yhä mukaan. Osallistu ircnetissä #skrolli tai lähesty sähköpostilla toimitus@skrolli.fi

Maksamme nykyään pieniä julkaisupalkkioita.

Tapaat meidät lokakuussa myös Skrollin ständillä WÄRK:festeillä sekä Alternative Partyilla. Tervetuloa tutustumaan ja ideoimaan yhdessä!



PDP-1

Hakkerikulttuurin juurilla

Olisiko 50 vuotta vanhan koneen bittejä kiva nyplätä? PDP-1 on tavoitettavissa emulaattorilla.

Teksti: Ville-Matias Heikkilä Kuvat: Ville-Matias Heikkilä, Wikimedia Commons (Joi Ito)

Mikä PDP-1?

Programmed Data Processor 1 oli yhdysvaltalaisen Digital Equipment Corporationin ensimmäinen tietokone ja merkittävä laite varhaisessa hakkerikulttuurissa. Sitä valmistettiin vuosina 1959-1969 melko vähän, mutta monien korkeakoulujen opiskelijat saattoivat varata konetta iltaisin käyttöönsä. Nämä etuoikeutetut pioneerit saivat ensimmäisinä tuntuman siihen, millaista tietotekninen harrastekulttuuri voi olla. Opiskelijat toteuttivat koneelle mm. kuuluisan Spacewar-videopelin sekä musiikki- ja grafiikkaohjelmia.

PDP-1 oli aikanaan suhteellisen pieni kone — painoa kokonaisuudella oli vain tonni, ja se mahtui pieneen huoneeseen. Keskusyksikön lisäksi peruskokoonpanoon kuuluivat tekstipäätteenä käytetty kaukokirjoitin, reikänauhan luku- ja lävistyslaitteet sekä iso, tutkanäyttöä muistuttava kuvaputki. Kone käytti edistyksellisesti transistoreja ja ferriittirengasmuistia, joten 1950-luvun tekniikalle tyypilliset pullonkaulat olivat sen osalta ollutta ja mennyttä.

PDP-1:n sananleveys on 18 bittiä ja keskusmuistia peruskokoonpanossa on 4096 sanaa, siis reilut 9 kilotavua. Muistin luku- ja kirjoitusoperaatiot vievät aikaa viisi mikrosekuntia, mikä on myös nopeimpien konekäskyjen suoritusaika. PDP-1 on siis laskentaominaisuuksiltaan verrattavissa 1980-luvun alun kotimikroihin.

Tietokoneharrastajat kokeilevat mielellään vanhoja laitealustoja, koska ne antavat kosketusta oman kulttuurimuodon juuriin ja ovat usein mukavan yksinkertaisia ja välittömiä. Toisaalta tietokoneharrastajilla on harvemmin kunnon käsitystä kasibittisaikakautta varhaisemmasta historiasta, joten Skrolli-lehti katsoo aiheelliseksi valottaa sitä.

Tässä artikkelissa lähestymme PDP-1:tä melko demoskenehenkisesti tarkoituksenamme ohjelmoida sille jotain, jota sillä ei ole koskaan aiemmin nähty. Jutussa esitetyt ideat ovat toki sovellettavissa muillakin alustoilla.

Emulaattori käyntiin

Tiettävästi enää yksi PDP-1 on käyttökunnossa. Kone sijaitsee tietokonehistoriallisessa museossa Kalifornian Mountain View'ssa. PDP-1:stä innostunut bittinikkari joutuu siis tyytymään emulaattoriin, ellei hän pääse hyviin väleihin museon henkilökunnan kanssa.

Muuhunkin kuin Spacewar-peliin taipuvia emulaattoreita on käytännössä tarjolla kaksi: vanhoihin mini- ja mainframekoneisiin keskittyvä SIMH ja kotitietotekniikkaan painottuva MESS. Koska SIMH ei tue grafiikkaa, ja koska haluamme tehdä etenkin grafiikkaohjelmia, valitsemme emulaattoriksi MESSin.

Kun MESS käynnistetään parametrilla "pdp1" ja varoitusdialogit ohitetaan, päästään näkymään, jonka oikeassa ylänurkassa on jäljitelmä PDP-1:n ohjauspaneelista kytkimineen. Vasemmassa ylänurkassa on musta tila, joka vastaa koneen kuvaputkea, ja alareunan valkoinen alue esittää paperipäätteen paperia.

Nykykoneista poiketen PDP-1 ei tee käynnistyttyään mitään. Siinä ei ole ROM-muistia, jonka sisältöä kone alkaisi suorittaa automaattisesti. Sen sijaan laitteessa on ohjauspaneeli, jolla voi käsitellä muistia bittitasolla ja käynnistellä ja pysäytellä ohjelman suoritusta. Read-in -kytkimellä kone alkaa suorittaa käskyjä reikänauhalta, joten valmisohjelmien lataamiseksi ei tarvitse nyplätä bittejä.

Ohjauspaneelia ei voi naksutella hiirellä, vaan se tapahtuu näppäimistöltä ctrl pohjassa. Ykkösellä ja Q:lla alkavat rivit kääntelevät osoitekentän kytkinten asentoja ja alemmat rivit testisanakentän. Näppäimet I:stä alkaen vastaavat paneelin kytkimiä

Spacewar-pelin ohjelmoijana tunnettu Steve Russell PDP-1:n ohjauspaneelin kimpussa.

run, stop, continue, examine ja deposit.

Testaamme emulaattoria aluksi lataamalla Spacewar-pelin RIM-tiedostosta, virtuaaliselta reikänauhalta. Kun tiedosto on mountattu valikon kautta, ctrl-enter käynnistää reikänauhanlukijan. Hyvin toimii, ja peli käynnistyykin automaattisesti, koska nauha loppuu sopivaan jmp-käskyyn.

Bitinnypläystä kytkimillä

Kun emulaattori on todettu toimivaksi, siirrymme syöttämään koneeseen omia ohjelmia aluksi suoraan ohjauspaneelista. Tässä korvaamaton apu on PDP-1:n käsikirja, joka löytyy helposti netistä.

PDP-1:n käskyt ovat yhden 18-bittisen sanan mittaisia. Ylemmät 6 bittiä on varattu käskykoodille ja alemmat 12 bittiä operandille, joka on tyypillisesti muistiosoite. PDP-1:n käskykanta on akkukeskeinen: useimmat käskyt operoivat muistipaikan ja akuksi kutsutun rekisterin välillä. Esimerkiksi käsky "add 1234" (oktaalina 401234, binäärinä 100-000-001-010-011-100) lisää akkurekisterin arvoon muistipaikassa 1234 olevan luvun. Seuraavalla aukeamalla on enemmän käskyesimerkkejä.

Haluamme luoda äkkiä jotain näkyvää. PDP-1:n näyttö on oskilloskooppityyppinen eli ampuu elektronisuihkuja määrättyihin kuvapinnan kohtiin. Konekäsky dpy ampuu näytölle pisteen paikkaan, jonka X-koordinaatti saadaan akusta ja Y-koordinaatti IO-rekisteristä. Nolla on ruudun keskellä. Laadimme minimaalisen kolmen sanan ohjelman (listaus 1), jonka pitäisi pommitella näytölle pisteitä kasvavilla akun arvoilla.

Naksutellessamme ohjelmaa muistiin huomaamme, ettei deposit-kytkin toimi lainkaan: sanat eivät yksinkertaisesti tallennu muistipaikkoihin. Pettymysten pettymys! Versiohistoria kertoo, että juuri edellisessä MESSin versiossa oli reikänauhanlukijakin rikki. Yritämme korjata toiminnallisuuden itse, mutta emulaattorin koodi ei meinaa kääntyä millään ja luovutamme. Olisimme halunneet saada edes jonkinlaisen tuntuman siihen, millaista varhaisten PDP-1-hakkerien harjoittama ruohonjuuritason kokeellisuus oli, mutta joudumme unohtamaan binäärikytkimet ja vilkkuvalot.

Kääntäminen reikänauhaksi

Kaivamme seuraavaksi esiin PDP-1-arkistointiprojektin FTP-palvelimelta esiin Digitalin alkuperäisen MACRO-assemblerin ja Extensive Typewriter-tekstieditorin. Emulaattorit eivät kuitenkaan oikein taivu niiden ajamiseen. Editoria ei saa nykyaikaiselta päätteeltä takaisin komentomoodiin, ja assembleri tuntuu jäävän aina jumiin riippumatta siitä, millaisia nauhoja sille syötetään. Unohdamme siis taas kerran autenttisuuden ja tyydymme SIMH:n mukana tulevaan macro1-ristiinkääntäjään.

Ohjelma kääntyy suoraan emulaattorissa käynnistyväksi RIM-tiedostoksi. Hienoa! Liikkuva piste toimii kuten pitääkin. Teemme siitä lukuisia muunnelmia ja kokeilemme siinä sivussa myös alkuperäistä munching squares -ohjelmaa (listaus 2).

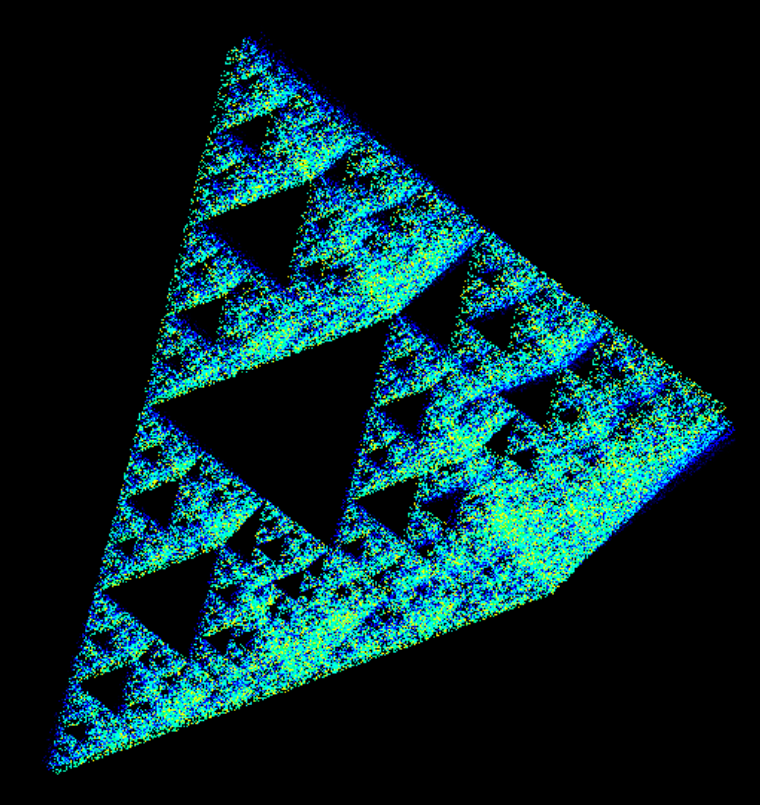
```
0  440003  idx 3
1  730007  dpy
2  600000  jmp 0
```

Listaus 1: kolmen käskyn grafiikkaohjelma oktaalina ja assemblyna

```
dpy=730007
lp, lat
    add v
    dac v
    rcl 9s
    xor v
    dpy
    jmp lp
v,  0
```

Listaus 2: munching squares -grafiikkaohjelma macro1:lle kelpaavana assemblyna

Pikkuohjelmien jälkeen on aika valita isomman projektin aihe. PDP-1:n ominaisuudet vaikuttaisivat erityisen sopivilta pisteistä koostuville grafiikkaefekteille. Miellyttävän leveiden sanojen ja kertolas-



Sierpinskin tetraedrin pyörittelyä C-kielisellä testiohjelmalla.

kunopeuden vuoksi harkitsemme etenkin 3D-pyöritystä ja fraktaaleja. Päädymme toteuttamaan pisteistä koostuvan, pyörivän Sierpinskin tetraedrin.

Toimiva malli ensin

Sierpinskin kolmio on yksinkertainen fraktaali, joka koostuu kolmesta kolmiosta, joista kukin koostuu samalla tavoin kolmesta kolmiosta ja niin edespäin. Sierpinskin tetraedri on kuvion kolmiulotteinen versio, jossa tetraedrit koostuvat tetraedreista. Toteutamme kuvion iteroituna funktiosysteeminä (IFS). Periaate on se, että uuden pisteen koordinaatit saadaan laskemalla keskiarvo edellisestä piirretystä pisteestä ja satunnaisesta kuvion ääripisteestä. Kuvio ei tule koskaan valmiiksi, joten pisteitä saa piirtää niin monta kuin suinkin ehtii.

Kun vaatimattomalla koneella toteutetaan vaativampia temppuja, on hyvä testata algoritmit ensin valmiiksi hieman korkeammalla abstraktiotasolla. Testaamme kuvion piirtämistä listauksen 3 mukaisella C-koodilla. Päätämme ottaa satunnaisluvut valmistaulukosta, jota sekoitetaan kunkin piirtorupeaman jälkeen hieman. Näin satunnaisuutta on riittävästi mutta ei liikaa.

```
drawSierpinskiDots()
{
  int i=0;

  for(;i<512;i++)
  {
    int r = rnd[i];
    x = (x+vertices[r+0])/2;
    y = (y+vertices[r+1])/2;
    dpy(x,y);
  }
}

makeRndTable()
{
  int a=0,i=0;
  for(;i<512;i++)
  {
    a>>=1;
    a^=i;
    a+=05624;
    rnd[i]=a&6;
  }
}

shuffleRnd()
{
  static int p=0;
  int i,j=p;
  for(i=0;i<15;i++)
  {
    rnd[j]^=6;
    j+=16;
  }
  p=(p+7)&15;
}
```

Listaus 3: C-malli kuvion piirtosilmukasta ja satunnaislukugeneroinnista.

Piirtovaiheessa meille riittävät kaksiulotteiset koordinaatit, sillä emme tarvitse perspektiivikorjausta emmekä taakse jäävien pisteiden piilotusta. 3D-pyöritykseen tarvitsemme kuitenkin kaikki komponentit.

Koska kuvion kaikki pisteet lasketaan neljästä kulmapisteestä, meidän tarvitsee pyörittää vain niitä. Tarkkuudesta ei siis tarvitse tinkiä nopeuden vuoksi. Pisteiden vähyydestä johtuen emme myöskään laske 3D-pyöritystä varten matriiseja, vaan hoidamme sen kolmen 2D-pyörityksen ketjuna. Käytämme 2D-pyöritysrutiinia myös sinitaulukon laskemiseen.

Kaikki laskenta tapahtuu PDP-1:ssä kokonaisluvuilla. Esitämme sinit ja kosinit 13072-osina, joten pudotamme kertolaskujen jälkeen 36-bittisestä tulosta alimmat 17 bittiä pois.

```
#define mul(a,b) ((((long long)
(a))*(b))>>17)

rotate2D(int*x,int*y,int s,int c)
{
  int x1 = mul(*x,c) + mul(*y,s);
  int y1 = mul(*y,c) - mul(*x,s);
  *x=x1; *y=y1;
}

// Osoittimet a, b ja c
viittaavat sinitaulukkoon,
// Sinin kaveriksi saa kosinin
samasta taulukosta
// 90 asteen eli 64
taulukkoaskeleen päästä.
rotate3D(int*x,int*y,int*z,int*a,
int*b,int*c)
{
  rotate2D(x,y,*a,*(a+64));
  rotate2D(x,z,*b,*(b+64));
  rotate2D(y,z,*c,*(c+64));
}

// Pyörityskulma vastaa ympyrän
256-osaa:
// 3197 = sin(2PI/256)*131072
// 131032 = cos(2PI/256)*131072
makeSineTable()
{
  int x=0,y=131071,i=0;
  for(;i<256+64;i++)
  {
    sine[i]=x;
    rotate2D(&x,&y,3197,131032);
  }
}
```

Listaus 4: 3D-pyörityksen C-malli sinitaulukointeineen. Koska testikoodi on hyvä pitää mahdollisimman lähellä kohdekoneen abstraktiotasoa, on siihen otettu mukaan myös muutama hyödyllinen osoitinkikkailu.

Ja sitten assemblerin kimppuun

Kun kuvio pyörii algoritmitestissä kuten pitääkin, on aika sovittaa koodi PDP-1-assemblyksi.

Koska ylivoimaisesti käytetyin koodin osa on yksittäisiä pisteitä iteroiva silmukka, panostamme siihen eniten. Yritämme saada siitä mahdollisimman nopean, ja rakennamme tarvittaessa koko muun ohjelman sen oikkujen varaan. Muutamia erilaisia vaihtoehtoja kokeiltuamme päädymme listauksen 5 mukaiseen koodiin.

```
loop,   law i 1000
        and xy
        sar 1s
ptr,    add i rndvtx
        dac xy

        rcr 9s
        sal 9s
        dpy

        idx ptr
        sas endptr
        jmp loop

        law rndvtx
        dap ptr
        jmp loop

xy,     0
endptr, add i rndvtx+1000
```

Listaus 5: Sierpinskin tetraedrin piirtosilmukka.

Silmukan alkuosa law-käskystä dac-käskyyn laskee uuden pisteen koordinaatit. Koska olemme maksimoineet koodin nopeuden, laskemme X- ja Y-komponentit samanaikaisesti. Otsakkeella xy merkityn muistipaikan yläpuolisko vastaa X:ää ja alapuolisko Y:tä. And-operaatiolla nollataan yksi välibitti, jotta puoliskot eivät sotkeentuisi toisiinsa. Laskennan jälkeen akun alapuolisko heitetään bittipyörityskäskyillä IO-rekisterin puolelle, josta dpy-käsky ottaa pisteen Y-koordinaatin.

Satunnaisten kulmapisteiden muistiosoitteilla täytetty rndvtx-taulukko käydään läpi itseäänmuuttavalla koodilla. Käsky idx kasvattaa muistipaikan ptr sisältöä eli vaihtaa siinä olevan epäsuoran add-käskyn viittaamaan seuraavaan muistiosoitteeseen. Sas-käsky vertaa uutta käskyä muistipaikassa endptr olevaan käskyyn ja hyppää jmp-käskyn yli, mikäli se on sama. Silmukan päätyttyä add-käsky palautetaan ennalleen law- ja dap-käskyillä.

Yksi silmukan kierros kestää käsikirjan luvuista summaamalla 135 mikrosekun-

Muutamia PDP-1:n käskyjä:

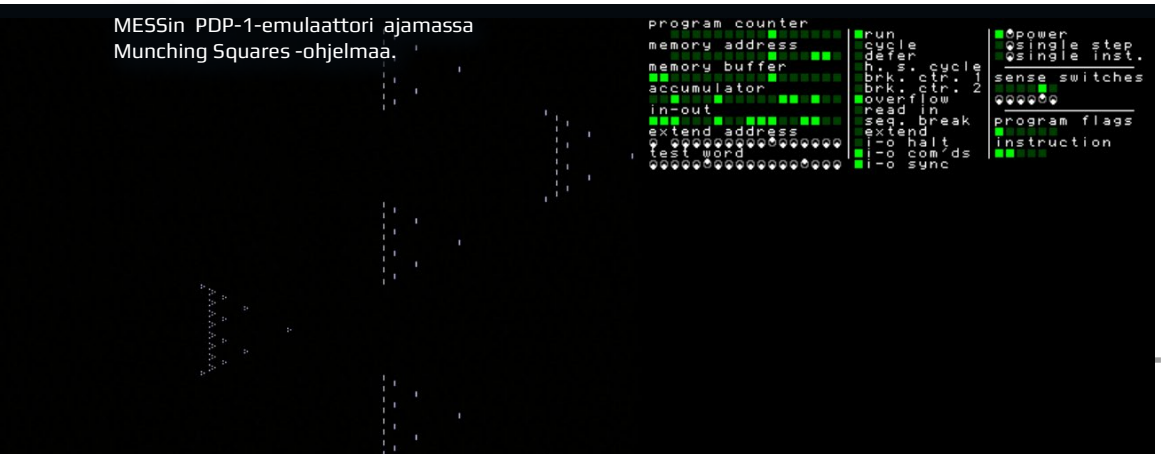
lac 1234	Lataa akkuun luvun muistipaikasta 1234. (Load ACcumulator)
lac i 1234	Epäsuora osoitus: lataa akkuun luvun muistipaikan 1234 osoittamasta muistipaikasta.
law 1234	Lataa akkuun luvun 1234. Luku ladataan negatiivisena, jos epäsuoruusbitti on päällä: law i 1234. (Load Accumulator With)
dac 1234	Tallentaa akun sisällön muistipaikkaan 1234. (Deposit ACcumulator)
dap 1234	Tallentaa akun 12 alinta bittiä muistipaikan 1234 vastaaviin bitteihin. (Deposit Address Part)
add 1234	Summaa akkuun muistipaikan 1234 arvon. Muita vastaavasti toimivia käskyjä ovat sub, and, xor ja ior.
idx 1234	Kasvattaa muistipaikan arvoa yhdellä ja lataa kasvatetun arvon akkuun. (InDeX)
sar 9s	Siirtää akun bittejä oikealle 9 askelta (Shift Accumulator Right). Sal siirtää vasemmalle. Sir ja sil pyörittävät IO-rekisteriä, scr ja scl akun ja IO-rekisterin yhdistelmää. Vastaavat r-alkuiset käskyt ovat kehäpyörityksiä (Rotate). 9s tarkoittaa lukua, jossa on 9 ykköstä (oktaalina 777).
jmp 1234	Hyppää muistipaikassa 1234 olevaan käskyyn. (JuMP)
sas 1234	Ohittaa seuraavan käskyn, jos akun arvo on sama kuin muistipaikan 1234 sisältö. (Skip if Accumulator Same)
dpy	Ampuu pisteen kuvaputkelle. X-koordinaatti akusta, Y-koordinaatti IO-rekisteristä. (DisPlaY)
lat	Lataa testisanakytkinten asennot akkuun. (Load Accumulator from Test word)

tia, mutta sen saisi hieman nopeammaksi unrollaamalla eli kirjoittamalla sisukset useampaan kertaan. Pidämme kuitenkin koodin lyhyenä, sillä saamme joka tapauksessa piirrettyä yli 7000 pistettä sekunnissa. Mikäli laskemme 512 pisteen rupeaman yhdeksi kuvaksi, saamme kuvataajuudeksi noin 14 fps. Ei mikään paha nopeus 50-luvun koneelta!

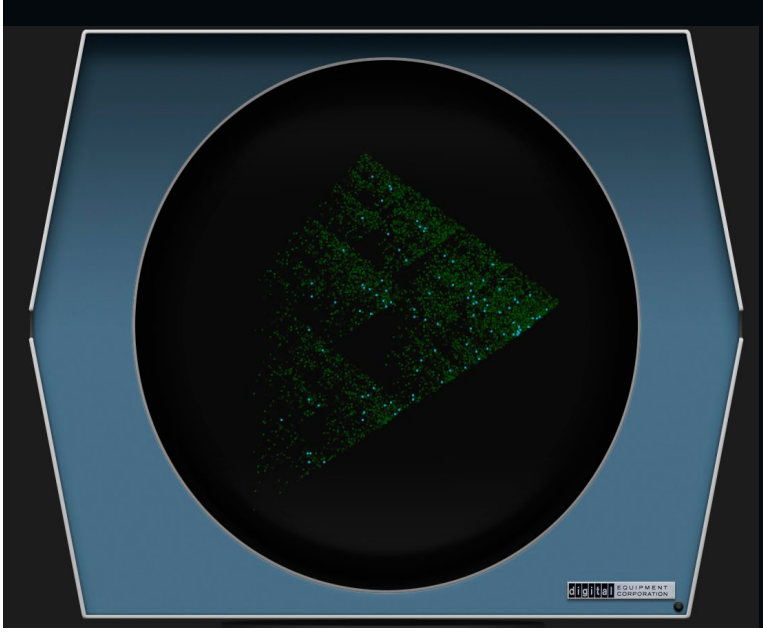
PDP-1-assemblyssa on paljon tuttua mikrotietokoneajan lapsille, mutta myös outoja vanhakantaisuuksia löytyy. Useimmilla käskyillä ainoat osoitusmuodot ovat suora ja epäsuora muistiviittaus. Rekistereillä ei voi osoittaa muistipaikkoihin, joten itseään muuttava koodi on usein paras tapa käydä läpi muistialueita. Ehdollisiin hyppykäskyihin ei voi antaa kohdeosoitetta, vaan ne hyppäävät aina seuraavan käskyn yli. Aritmetiikka ei tunne muistinumerolippua. Negatiiviset luvut esitetään yhden komplementtina, eli lukujärjestelmään kuuluu sekä negatiivinen että positiivinen nolla. Aliohjelman on ensi töikseen korjattava lopussaan oleva jmp-käsky osoittamaan oikeaan paluuosoitteeseen.

Häkkyrä pyörimään

Kun ohjelma toimii hyvin kiinteillä kulmapistekoordinaateilla, pääsemme toteuttamaan 3D-pyörityksen. Noudatamme melko suoraan C-mallia ilman kummempia lisäkikkailuja. Kertolaskurutiinin pääsemme toteuttamaan



MESSin PDP-1-emulaattori ajamassa Munching Squares -ohjelmaa.



Lopullinen fraktaaliohjelma sovitettuna Masswerk.at:stä nyysittyyn emulaattoriin.

itse, sillä PDP-1:n perusmallissa on mul-käskyn sijaan pelkkä yhden kertolaskuaskelen hoitava mus-käsky.

Kun kuvio pyörii kuten pitääkin, on viimeistelyn aika. Kehitysvaiheessa käytetyt parametrit kannattaa vaihtaa sellaisiksi, joilla lopputulos näyttää mahdollisimman hyvältä. Montako pistettä kannattaa piirtää pyöritysten välissä? Millaiset pyöritysaskelet ovat hyvät? Kuinka satunnaislukutaulukkoa kannattaa sekoittaa?

Viimeisteltyä tetraedriohjelmaa pyörittävä emulaattori ja linkki täydelliseen lähdekoodipakettiin löytyvät sivultamme osoitteesta http://skrolli.fi/pdp1 .

Mitä kokemuksesta kostui?

Monille suosituille koneille on hyviä emulaattoreita, joilla kaikki toimii suoraan, ja jotka kelpaavat tarkkuutensa puolesta jopa rautakikkailudemojen tekemiseen. PDP-1-emulaattorit ovat kuitenkin nykyisellään hyvin vaikeita, keskeneräisiä ja bugisiakin. PDP-1:een tutustuminen MESSin kautta vaatii nykyisellään jonkin verran salapoliisityötä, ja minkäänlaista tuntumaa bittikytkimiin tai reikänauhoihin on liki mahdoton saada.

Assembler-ohjelmoinnin kautta syntyi kuitenkin jonkinlainen suhde tähän "maailman ensimmäiseen lelutietokoneeseen". PDP-1 on konekielitasolla melko helposti lähestyttävä kone, ja sitä voisi yksinkertaisuutensa vuoksi jopa suositella tietokonetekniikan tai assemblyohjelmoinnin alkeiskursseille. Konekieliohjelmointi on edelleen melko samankaltaista kuin puoli vuosisataa sitten. Monet asiat ovat toki muuttuneet, mutta yllättävän monet ovat myös pysyneet.

Nykykulttuuria ei voi ymmärtää ilman historiaa. Tästä syystä PDP-1:n kaltaiset hakkerihistoriallisesti merkittävät koneet ansaitsisivat arvoisensa emulaation, joka vangitsisi niiden teknisen olemuksen ja käyttökokemuksen mahdollisimman hyvin. Vanhoja pelejä arkistoidaan ja emuloidaan jo melkoisen kiitettävästi, mutta aika alkaisi olla kypsä myös laajemmalle tietokonehistorian elävöittämisliikkeelle. Alkuperäiset PDP-1-hakkerit kuolevat jo vanhuuteen, eikä viimeinen toimiva laitekaan ikuisesti pysy käyttökunnossa. Kuka lähtisi ajamaan asiaa?

Ei näin!

Apple Pippin, iFail

Apple on viime vuosina tullut tunnetuksi rohkeista ja uutta luovista tuote-esittelyistään. Yritys tehdä Macintoshista pelikonsoli ei ollut oikein kumpaakaan.

Teksti: Mikko Heinonen Kuva: Manu Pärssinen

Apple Computer oli hukassa 1990-luvun alkupuolella. Puolijumala Steve Jobs oli lähtenyt firmasta vuonna 1985 riitaannuttuaan hallituksen kanssa, kehittänyt NeXT-mikron ja perustanut Pixar-elokuvastudion.

Välirikosta lähtien Apple oli yrittänyt keksiä seuraavaa isoa juttuaan. Yhtiön tietokonemallisto alkoi olla jo vanhahtavaa, vaikka PowerBook-kannettavan esittely olikin tuonut myynteihin tilapäistä nostetta. Microsoft Windows nakersi Applen markkinaosuuksia myös graafiseen käyttöön tarkoitettujen työasemien markkinoilla.

Toistaiseksi Applella oli onnistuttu vain toteamaan, ettei uusi menestysresepti ollut ainakaan Newton-kämmenmikro, uusi A/UX-käyttöjärjestelmä, digitaalikamera tai Philipsin hifi-laitteiden myynti omalla nimellä.

Auttava käsi idästä

Toisen maailmansodan jälkeen pienoismallien teon aloittanut Bandai oli jo 1980-luvulla vakiinnuttanut asemansa yhtenä Japanin ja koko maailman suurimmista leluyhtiöistä. Myös sähköinen viihde oli Bandain sydäntä lähellä. Niinpä 90-luvun alussa se halusi synnyttää uuden, monipuolisemman laitteen CD-ROM-pelien pelaamiseen. Ennestään sillä oli jo Playdia, yksinkertainen CD-konsoli visailu- ja anime-peleille.

Arvioituaan markkinoita Bandai tuli siihen tulokseen, että Apple Macintosh sisältää valmiiksi halutut ominaisuudet. Pitäisi siis vain suostutella Apple suunnittelemaan kotimikrostaan laihdutettu versio, jota Bandai voisi valmistaa lisenssillä.

Kun Bandai vuonna 1994 lähestyi Applea asian tiimoilta, se oli kiinnostunut ennen kaikkea Motorolan 68030-prosessoriin pohjautuvasta Macintoshista. Valmiina oli jopa karkea prototyyppi. Tämä lienee ollut Applelle miettimisen paikka. Pelikonsoli saattaisi tuoda kassaan kaivattua täytettä, mutta toisaalta Apple oli juuri siirtymässä voimakkaasti kohti uutta PowerPC-arkkitehtuuria.

> *Peli-Macintosh ei pärjännyt Sonyn, Nintendon ja kumppaneiden konsoleille.*

Lopulta asian ratkaisi se, että tutkimusten mukaan kuluttajat halusivat laitteeseen mahdollisuudet tietoverkkojen käyttöön. Koska Apple oli juuri pari vuotta aiemmin uudistanut tavan, jolla Mac käytti modeemia, käytännössä ainoa mielekäs vaihtoehto oli suunnitella koko laite uudelleen PowerPC 603 -prosessorin ympärille.

Omppuja olkkariin

Pippin ei saanut nimeään suinkaan Keski-Maan tarustosta, vaan Newport Pippin -omenasta. Nimi olikin kuvaava: sen sijaan, että Apple olisi vain myynyt Bandaille konsoliraudan, se halusi tehdä Pippinistä uuden omenalajikkeen, koko kodin viihdealustan.

PowerPC:n ympärille suunniteltu Pippin perustui kevennettyyn System 7.5.2 -käyttöjärjestelmään, maksimissaan 32 megatavun muistiin ja CD-asemaan.

Ajoittaisen realismin puuskassa Apple ei ajatellut ryhtyvänsä itse konsolivalmistajaksi, vaan myyvänsä lisenssejä laitteiston valmistukseen eri yhtiöille. Koska Pippin oli sisäisesti Mac, kasvattaisi se näin välillisesti Macintoshinkin markkinaosuutta. Samaan päämäärään Apple oli pyrkinyt jo muutenkin sallimalla klooni-Macien valmistuksen. Lopulta Bandain lisäksi lisenssin hankki vain yksi yritys, Katz Media, joka sekin valmistutti koneensa Bandailla.

Koska konsoli käytännössä perustui Macintosh-laitteistoon, varsinainen suunnitteluvaihe ei kestänyt kovin kauaa. Bandai Pippin Atmark tuotiin Japanin markkinoille jo vuoden 1995 maaliskuus-

sa. Miltei 65 000 jenin hintaisena se oli tyyris, sillä esimerkiksi Sony PlayStation maksoi julkaisussa alle 40 000 jeniä.

Kalliissa laitteessa oli toki monenlaisia ominaisuuksia. Modeemin lisäksi Pippin sisälsi muun muassa tulostin- ja näppäimistöliitännät ja jopa lähdön VGA-näytölle. Käytännössä se oli vain vähän riisuttu tietokone. Vakiomallissa oli muistia peräti kuusi megatavua (PlayStationissa kaksi Mt) ja lisämuistiakin oli kaupan. Peliohjaimeen oli yhdistetty trackball, jolla saattoi käyttää hiiren vaativia ohjelmistoja.

Pohjois-Amerikassa ja Euroopassa laite sai nimekseen Bandai Pippin @WORLD. Se julkaistiin vasta vuoden 1996 syksyllä, vaikkei eroina ollut kuin käyttöjärjestelmän kieli ja kotelon väri. Länsimaihin tarkoitetut Pippinit olivat mustia, jutun kuvassa esiintyvä valkoinen kone on Japanin malli.

Kuka tätä tarvitsee?

Pelikonsolien maailmassa miljoonan kappaleen myynti on lähinnä hyvä alku, ja kunkin sukupolven voittajista voidaan puhua kun myyntiluvut pyörivät kymmenissä miljoonissa. Bandai ei tähdännyt Pippinillä taivaisiin, vaan suunnitteli myyvänsä niitä noin puoli miljoonaa vuodessa. Tähänkään ei päästy, vaan ennen laitteen tuotannon lopettamista vuonna 1997 niitä ehdittiin saada kaupaksi noin 40 000 kappaletta.

Jos menekki Japanissa oli laiskanlaista, lännessä Pippiniä ei ostanut käytännössä kukaan. Eri lähteissä mainitaan vain noin 5 000 kappaleen myynti, ennen kuin Bandai laivasi myymättömät koneet takaisin Japaniin. Tämä musta versio laitteesta onkin huomattavan harvinainen, vaikkei valkoistakaan voi yleiseksi kutsua.

Epäonnistumisen syitä on melkein valittavaksi asti. Paitsi, että Pippin oli todella kallis (USA:ssakin 600 dollaria), se oli jo lähtökohtaisesti sopimaton pelikoneeksi. Vaikka laite oli tehokas, kokonainen tietokoneen käyttöjärjestelmä vei suuren osan resursseista.

Macintosh-variantti ei pystynyt kilpailemaan pelisuorituskyvyssä Sonyn, Nintendon ja kumppaneiden erikoisrautaa sisältäneitä pelikonsoleita vastaan. Valtaosa vähäisistä nimikkeistä oli kankeaa multimediaa, harvat varsinaiset pelit käännöksiä Macilta. Pelaajalle syitä Pippinin hankintaan ei oikein ollut.

Näin Pippin tavoitteli lähinnä kuluttajia, jotka halusivat mitenkuten pelikelpoisen nettipäätteen, mutta eivät oikeaa tietokonetta. Heitä ei ollut likimainkaan tarpeeksi. Katz Media solmi kyllä sopimuksia Pippin-laitteiden myynnistä hotelleihin ja infokioskeiksi, mutta nämä ovat kuluttajakauppaan verrattuna pieniä toimialoja. Pelijulkaisijat olisivat kaivanneet todellista volyymia.

Bandaita Pippinin epäonnistuminen tuskin paljon heilutti. Se kun ei ollut edes ainoa pelikonsoli, jota yhtiö valmisti ja myi – lelumyynnistä nyt puhumattakaan. Ja vaikkei Applenkaan tulevaisuus sentään Pippinin varassa ollut, jatkettiin alustan kehittelyä sen floppaamisesta huolimatta. Työn lopetti vasta Steve Jobs, joka Applen ruoriin palattuaan katkoi kerralla kaikki Mac-tuoteperheen ulkopuoliset versot.

Vektorista kvaternioon

3D-matematiikan peruspalikat

Vahva perusta on olennaista 3d-grafiikan hallitsemiselle. Pohjana olevien matemaattisten käsitteiden osaaminen auttaa ymmärtämään mitä pinnan alla tapahtuu.

Teksti: Mikko Rasa Kuvat: Mikko Rasa, Mitol Berschewsky

Vektorit

Tärkein 3D-grafiikassa käytettävä matemaattinen käsite on vektori. Muodollisesti se on järjestetty joukko lukuja. Kun luvut tulkitaan koordinaattiakselien suuntaisina siirtyminä, vektori esittää kahden pisteen välistä siirtymää. Toisin ilmaistuna vektorilla on suunta ja pituus. Jos ensimmäinen piste sijoitetaan origoon, vektori kuvaa toisen pisteen sijaintia. Jos vektorin kaikki alkiot ovat nollia, sitä kutsutaan nollavektoriksi.

Vektorien esitystavat vaihtelevat hieman, mutta kaikille niille on yhteistä jonkinlaiset sulut, joiden sisällä vektorin alkiot ovat. Vektorimuuttuja on yleensä pienaakkonen, joka on joko lihavoitu tai sen päälle on piirretty nuoli. Tässä artikkelissa käytämme hakasulkuja ja nuolimerkintää.

$$\vec{v} = [1\ 2\ 0]$$

Vektorin voi esittää myös vektoriavaruuden kantavektorien summana. Kantavektoreita on yhtä monta kuin avaruudella ulottuvuuksia, ja kukin kantavektori on tietyn ulottuvuuden suuntainen yksikkövektori. Euklidisen avaruuden kantavektorit ovat koordinaattiakselien suuntaiset yksikkövektorit. Kolmiulotteisen avaruuden kantavektoreita merkitään yleensä kirjaimilla $\vec{\imath}$, $\vec{\jmath}$ ja $\vec{k}$.

$$\vec{v} = \vec{\imath}+2\vec{\jmath}$$

Eräs vektorin perusoperaatio on normi, joka euklidisessa avaruudessa on sama kuin sen pituus. Normia merkitään kaksoisviivoilla vektorin molemmin puolin.

Vektoreita voi laskea yhteen ja vähentää toisistaan. Tällöin kummankin vektorin vastaaville alkioille suoritetaan erikseen yhteen- tai vähennyslasku. Vastaavalla tavalla vektorin voi myös kertoa tai jakaa skalaarilla eli paljaalla luvulla.

$$[1\ 2\ 0] + [1\ 0\ 3] = [2\ 2\ 3]$$
$$[2\ 2\ 2] - [4\ 2\ 0] = [-2\ 0\ 2]$$
$$[2\ 1\ 3] \cdot 1.5 = [3\ 1.5\ 4.5]$$

Tavanomaista kertolaskuoperaatiota kahden vektorin välille sen sijaan ei ole määritelty. Tarjolla on kuitenkin kaksi erilaista kertolaskua muistuttavaa operaatiota. Ensimmäinen on skalaaritulo eli pistetulo (eng. dot product), joka nimensä mukaisesti antaa tulokseksi skalaarin. Se muodostetaan kertomalla vektorien vastaavat alkiot keskenään ja laskemalla tulot yhteen. Nimitys pistetulo taas johtuu merkintätavasta, joka on rivin keskellä oleva piste.

$$[1\ 2\ 1] \cdot [1\ 0\ 3] = 1\cdot1 + 2\cdot0 + 1\cdot3 = 4$$

Pistetulo on suuruudeltaan sama kuin vektorien pituuksien tulo kerrottuna niiden välisen kulman kosinilla: $\vec{v}\cdot\vec{u} = \|\vec{v}\|\|\vec{u}\|\cos(\Theta)$. Tästä seuraa joitakin hyödyllisiä ominaisuuksia. Vektorin pistetulo itsensä kanssa on sen pituuden neliö: $\vec{v}\cdot\vec{v} = \|\vec{v}\|^2$. Kahden vektorin välinen pistetulo on nolla, jos ja vain jos vektorit ovat kohtisuoria ($\cos(\Theta)=0$) tai toinen vektoreista on nollavektori ($\|\vec{v}\|=0$ tai $\|\vec{u}\|=0$). Pistetulo on positiivinen, jos vektorien välinen kulma on terävä, ja negatiivinen, jos kulma on tylppä. Kahden yksikkövektorin pistetulosta saadaan arkuskosinilla niiden välinen kulma. Monissa yhteyksissä tosin tarvitaan nimenomaan kosinia.

Toinen vektorien välinen operaatio on vektoritulo eli ristitulo. Se on määritelty ainoastaan kolmiulotteisille vektoreille ja antaa tuloksena uuden vektorin. Ristitulo lasketaan determinanttina matriisista, jonka ensimmäisellä rivillä ovat kantavektorit, toisella rivillä tulon vasen vektori ja kolmannella rivillä oikea vektori. Merkintätapa on vinoristi.

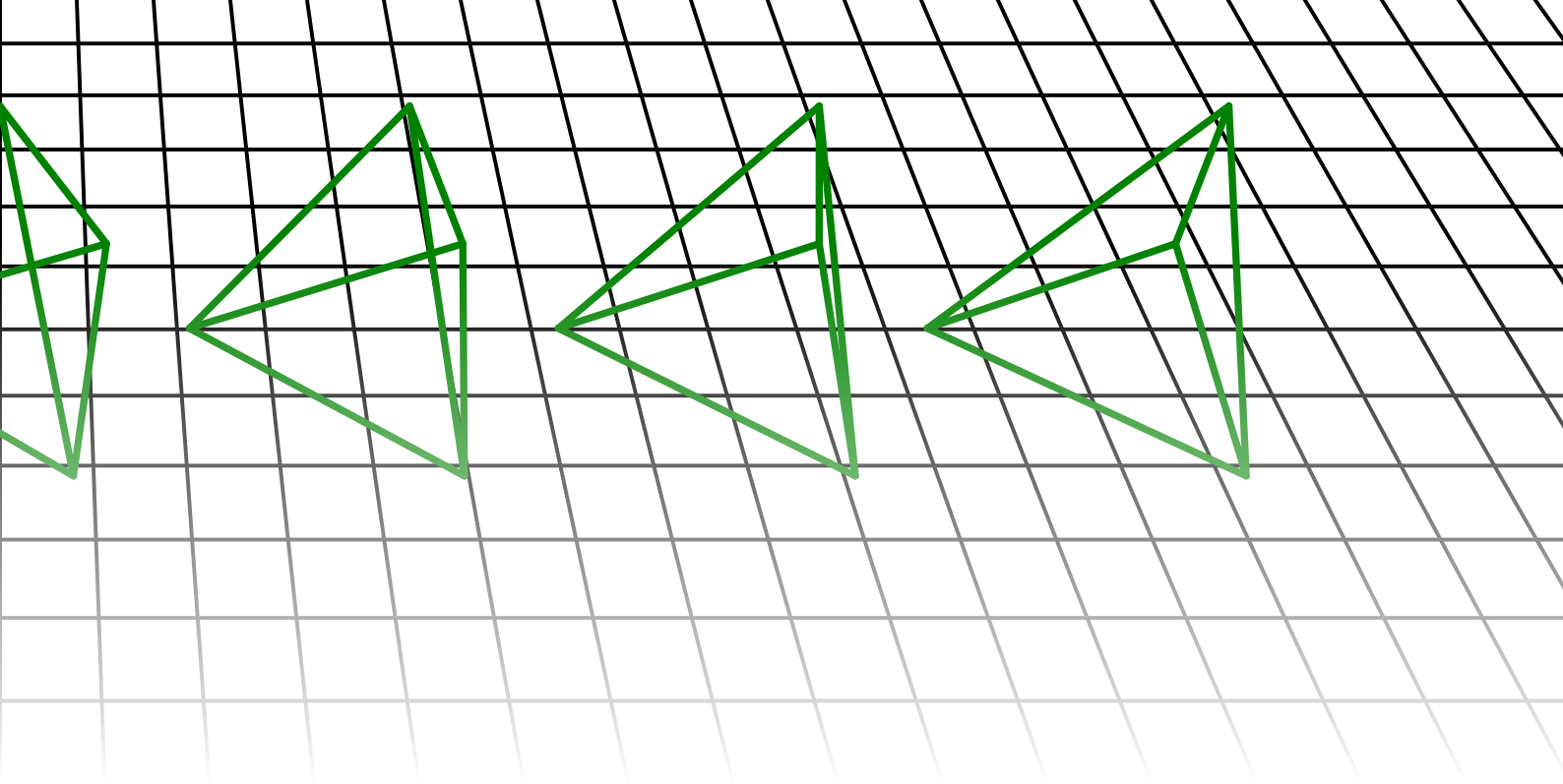
$$[1\ 1\ 0] \times [-1\ 0\ 1] =$$
$$\det\left(\begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}\right) = (1\cdot1-0\cdot0)\mathbf{i} + (0\cdot(-1)-1\cdot1)\mathbf{j} + (1\cdot0-1\cdot(-1))\mathbf{k} = [1\ -1\ 1]$$

Toisin kuin pistetulo, ristitulo ei ole vaihdannainen. Jos operandien järjestys vaihdetaan, tulovektori muuttuu vastakkaissuuntaiseksi.

Myös ristitulolla on muutama hyödyllinen ominaisuus. Tulovektori on aina kohtisuorassa molempia operandeja kohtaan. Sen pituus on operandien pituuksien tulo kerrottuna niiden välisen kulman sinillä. Ristitulo on nollavektori, jos ja vain jos operandit ovat saman- tai vastakkaissuuntaisia tai toinen niistä on nollavektori. Käytettäessä tulovektoria pyöritysakselina on kulma ensimmäisestä operandista toiseen positiivinen. Euklidisen avaruuden kantavektoreille pätee $\vec{\imath}\times\vec{\jmath} = \vec{k}$, $\vec{\jmath}\times\vec{k} = \vec{\imath}$ ja $\vec{k}\times\vec{\imath} = \vec{\jmath}$.

Matriisit

Toinen tärkeä käsite on matriisi. Se on



kaksiulotteinen taulukko lukuja. Myös vektori voidaan ajatella matriisina, jonka toinen dimensio on yksi. Näin saadaan joko rivivektori tai sarakevektori. Matriisin puolestaan voidaan ajatella koostuvan rivi- tai sarakevektoreista.

Matriisia merkitään hakasuluilla, joiden välissä matriisin alkiot ovat. Hakasulut ovat koko matriisin korkuiset. Jos pystysuuntaista tilaa on niukasti, voidaan myös käyttää vektorimerkintää, jonka alkiot ovat vektoreita. Matriisimuuttuja on yleensä lihavoitu suuraakkonen. Matriisin kokoa ilmoitettaessa mainitaan rivimäärä ensin. Esimerkiksi 3×2 alkion matriisissa on kolme riviä ja kaksi saraketta. Jos rivien ja sarakkeiden määrä on sama, sanotaan kyseessä olevan neliömatriisi.

$$\mathbf{M} = \begin{bmatrix} 1 & 3 & 2 \\ 3 & 2 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Matriisien yhteen- ja vähennyslasku sekä kerto- ja jakolasku skalaarin kanssa toimii samoin kuin vektoreilla, eli operaatio suoritetaan jokaiselle elementille erikseen.

Matriiseille on määritelty kertolasku, mutta vain jos vasemmanpuoleisessa matriisissa on yhtä monta saraketta kuin oikeanpuoleisessa on rivejä. Lopputuloksessa on yhtä monta riviä kuin vasemmanpuoleisessa matriisissa ja yhtä monta saraketta kuin oikeanpuoleisessa. Myöskään matriisien kertolasku ei ole kommutatiivinen. Tulosmatriisin elementti i,j on määritelty vasemmanpuoleisen matriisin rivivektorin ja oikeanpuoleisen matriisin sarakevektorin pistetulona. Matemaattisena kaavana ilmaistuna $C_{i,j} = \sum_{k=0}^{n} A_{i,k} B_{k,j}$.

$$\begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} & \mathbf{A}_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \times \begin{bmatrix} \mathbf{B}_{1,1} & B_{1,2} & B_{1,3} \\ \mathbf{B}_{2,1} & B_{2,2} & B_{2,3} \\ \mathbf{B}_{3,1} & B_{3,2} & B_{3,3} \end{bmatrix} = \begin{bmatrix} \mathbf{C}_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,1} & C_{3,2} & C_{3,3} \end{bmatrix}$$

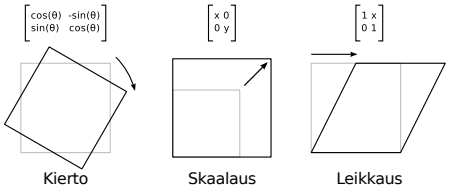
$$\mathbf{C}_{1,1} = \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1} + \mathbf{A}_{1,3}\mathbf{B}_{3,1}$$

Muunnokset

Koska vektori voidaan esittää yksirivisenä tai -sarakkeisena matriisina, vektori ja matriisi on mahdollista kertoa keskenään. Tuloksena on uusi vektori. Neliömatriisilla kerrottaessa alkioiden määrä säilyy samana. Tällaista operaatiota kutsutaan lineaarimuunnokseksi, koska saadun vektorin alkiot muodostetaan alkuperäisen vektorin alkioista vakiokertoimilla, eli lineaarikombinaationa.

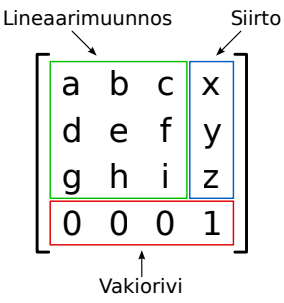
Lineaarimuunnoksista on eroteltavissa kolme perusoperaatiota: kierto (eng. rotation), skaalaus (eng. scaling) ja leikkaus (eng. shear). Mikä tahansa lineaarimuunnos on hajotettavissa näiden yhdistelmäksi.

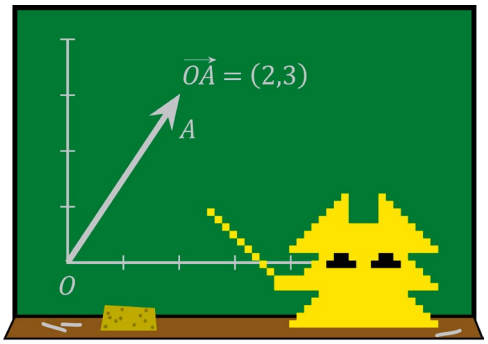
Lineaarimuunnoksen voi myös ajatella kuvaavan kahden vektoriavaruuden välistä suhdetta. Tällöin muunnosmatriisin sarake- tai rivivektorit ovat lähtöavaruuden kantavektorit esitettynä kohdeavaruudessa. Muunnos suoritetaan kertomalla kukin kantavektori vastaavalla lähtövektorin komponentilla.



Tarkkasilmäinen lukija saattoi havaita, että eräs tuiki tarpeellinen operaatio puuttuu äskeisestä listasta. Lineaarimuunnoksilla ei ole mahdollista toteuttaa siirtoa (eng. translation). Siirrossa vektorin kuhunkin alkioon pitäisi lisätä jokin vakioarvo. Avuksi otetaan affiinimuunnos. Siinä avaruuteen lisätään keinotekoinen ulottuvuus, jota kuvaavan alkion arvo on useimmiten 1. Nyt siirto voidaan toteuttaa leikkauksena kohtisuoraan tätä ulottuvuutta vastaan.

Affiinimuunnos kolmiulotteisessa avaruudessa on 4×4 matriisi. Vasemman yläkulman 3×3 osamatriisi on lineaarinen muunnos. Oikeanpuolimmaisen sarakkeen kolme ensimmäistä alkiota kuvaavat siirtoa. Alimman rivin sisältö on aina [0 0 0 1].





Kvaterniot

Vähemmän tunnettu mutta silti hyödyllinen käsite on kvaternio (eng. quaternion). Se on neliulotteinen luku, vastaavasti kuin kompleksiluku on kaksiulotteinen. Kvaterniolla on kolme erillistä imaginääriosaa, joita usein käsitellään kolmiulotteisena vektoriosana.

Kvaternio voidaan esittää vektorin tapaan järjestettynä lukujoukkona. Kvaternion ympärillä käytetään tavallisesti kaarisulkuja. Toinen mahdollinen merkintätapa on polynomi, jossa imaginääriyksiköitä merkitään kirjaimilla i, j ja k.

p = (1 0 2 3)
p = 1+2j+3k

Imaginääriyksiköiden kertolaskukaavat:
ij = -ji = k
ik = -ki = -j
jk = -kj = i

Kvaternioiden laskusäännöt voidaan johtaa kaavasta $i^2 = j^2 = k^2 = ijk = -1$. Kertolasku ei ole kommutatiivinen.

Yhteen- ja vähennyslasku hoituvat suorittamalla operaatio erikseen kullekin komponentille. Myös kertolasku on polynomimuodon perusteella suoraviivainen, joskin pitkähkö.

Kvaternioiden kertolasku:
$(a_1+b_1i+c_1j+d_1k)(a_2+b_2i+c_2j+d_2k) =$
$(a_1a_2-b_1b_2-c_1c_2-d_1d_2) +$
$(a_1b_2+b_1a_2+c_1d_2-d_1c_2)i +$
$(a_1c_2+c_1a_2-b_1d_2+d_1b_2)j +$
$(a_1d_2+d_1a_2+b_1c_2-c_1b_2)k$

Kompleksilukujen tapaan kvaternioille on määritelty konjugaatio, jossa imaginääriosien etumerkit muutetaan päinvastaisiksi. Konjugaatiota merkitään muuttujan jälkeen tulevalla tähdellä.

3D-matematiikan kannalta kvaternioista tekee kiinnostavia niiden kyky esittää 3D-avaruuden kiertoa. Jos yksikkökvaternio kirjoitetaan muodossa $\cos(\theta/2)+\sin(\theta/2)(xi+yj+zk)$, sillä voidaan kiertää vektoria kulman θ verran akselin [x y z] ympäri. Itse operaatio on hieman monimutkainen. Vektorista on ensin tehtävä kvaternio lisäämällä siihen nolla-arvoinen reaaliosa. Sen jälkeen se kerrotaan vasemmalta kiertokvaterniolla ja oikealta sen konjugaatilla. Lopuksi reaaliosa poistetaan, ja jäljelle jää kierretty vektori.

Kiertokvaternioilla on monia hyödyllisiä ominaisuuksia muihin kiertojen esityksiin verrattuna. Kulmiin perustuvassa esityksessä esiintyviä vapausastelukkiutumia ei ole, vaan mielivaltaisten kiertojen ketjutus on helppoa. Etenkin vanhoissa peleissä voi joskus nähdä, kun lähes pystysuorassa oleva objekti pyörii kummallisesti jouduttuaan lähelle lukkiutumaa.

Matriisiin verrattuna kvaternion etuna on virheensietokyky. Koska kierrolla on kolme vapausastetta ja kiertomatriisissa yhdeksän alkiota, ei ole yksiselitteistä tapaa normalisoida virheellistä matriisia. Kvaterniolle sen sijaan riittää palauttaminen yksikköpituuteen. Tämä on olennaista, koska muunnoksia ketjutettaessa pienetkin virheet kasautuvat.

Haittapuolena kvaternioissa on kierron suorittamisen monimutkaisuus. Tarvittavien alkeisoperaatioiden määrä on kaksinkertainen matriisiin verrattuna. Kvaterniolla ei pysty esittämään mitään muuta kuin kierron, joten esimerkiksi siirto täytyy tallentaa erikseen. Myöskään 3D-rajapinnat eivät tue kvaternioita. Usein on järkevää käyttää kvaterniota simulaation pyörittämiseen ja laskea sen perusteella matriisi grafiikkaa varten.

Soveltaminen ohjelmointiin

Vektorin tallentaminen tietokoneen muistiin on helppoa. Se on yksinkertaisesti kiinteän mittainen lukutaulukko. Kolmiulotteisiin vektoreihin lisätään joskus neljäs alkio, jonka arvo on 1. Se helpottaa affiinien muunnosten laskemista ja prosessorin SIMD-käskyjen käyttöä. Hintana on tarvittavan tilan kasvaminen kolmanneksella.

Matriisi voidaan tallentaa yksi- tai kaksiulotteisena taulukkona. Kummassakin tapauksessa on päätettävä, onko tallennustapa riveittäin vai sarakkeittain, ja suoritetaanko muunnokset rivi- vai sarakevektoreilla. Omia matriisirutiineja koodatessa on tärkeää sisäistää näiden valintojen merkitys. Muuten seurauksena voi olla outoja bugeja, joiden selvittäminen voi olla vaikeaa.

OpenGL:ssä matriisit esitetään sarakkeittain ja käytetään sarakevektoreita matriisin oikealla puolella. Direct3D toimii molemmissa suhteissa päinvastoin.

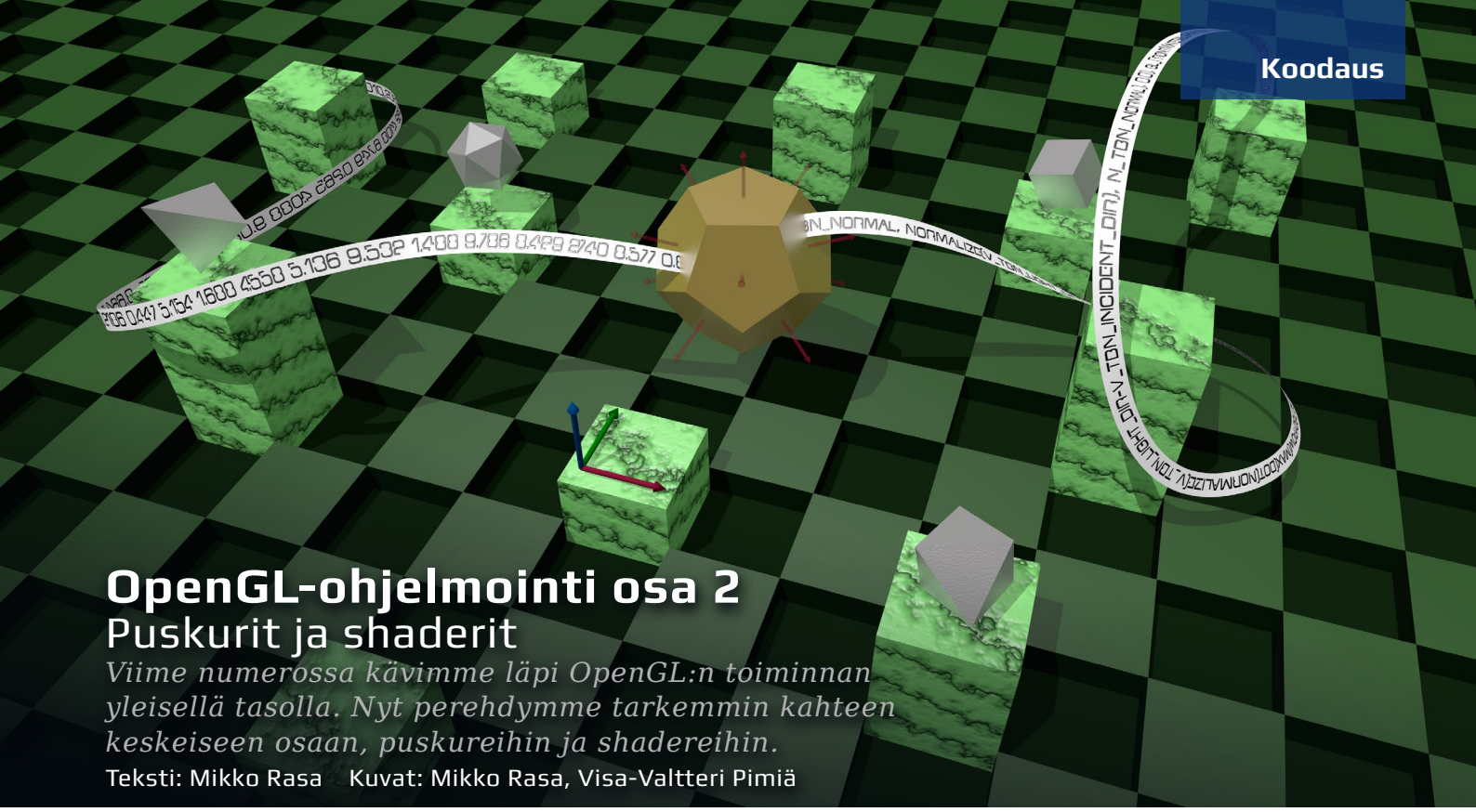
Kvaternio tallennetaan samaan tapaan kuin neliulotteinen vektori. Reaaliosan voi sijoittaa joko ensimmäiseen tai viimeiseen alkioon.

Myös käytettävän tietotyypin tarkkuuteen on syytä kiinnittää huomiota. Jos ohjelmassa esiintyy suuria suhteita etäisyyksien välillä, 32-bittisen liukuluvun dynaaminen tarkkuus ei välttämättä riitä. Tämän huomaa grafiikassa liikkeiden nykimisenä tai geometrian palikoitumisena. Toisaalta 64-bittiset liukuluvut vaativat kaksinkertaisen määrän tilaa, joten niitä ei kannata sokeasti käyttää kaikkialla.

Valmiitakin matematiikkakirjastoja on saatavilla, ja useimmissa korkean tason 3D-kirjastoissa on omat matematiikkarutiininsa. Näitä käytettäessä on syytä perehtyä kirjaston dokumentaatioon ja käytäntöihin ongelmien välttämiseksi. Jos matematiikka- ja grafiikkakirjastot ovat erilliset, voi olla tarpeen kirjoittaa muunnosrutiinit niiden välille.

Pseudokooditoteutukset eräistä vektori- ja matriisioperaatioista:

```
function length: vector3 a -> number
  return sqrt(a.x*a.x + a.y*a.y + a.z*a.z)

function dot_product: vector3 a, vector3 b -> number
  return a.x*b.x + a.y*b.y + a.z*b.z

function cross_product: vector3 a, vector3 b -> vector3
  return vector3(a.y*b.z-a.z*b.y, a.z*b.x-a.x*b.z, a.x*b.y-a.y*b.x)

function matrix_product: matrix3 a, matrix3 b -> matrix3 result
  for i in 1..3
    for j in 1..3
      for k in 1..3
        result[i, j] = a[i, k]*b[k, j]
```



OpenGL-ohjelmointi osa 2

Puskurit ja shaderit

Viime numerossa kävimme läpi OpenGL:n toiminnan yleisellä tasolla. Nyt perehdymme tarkemmin kahteen keskeiseen osaan, puskureihin ja shadereihin.

Teksti: Mikko Rasa Kuvat: Mikko Rasa, Visa-Valtteri Pimiä

OpenGL:n oliot

OpenGL on pohjimmiltaan oliopohjainen rajapinta. Varsinaiset oliot ovat kuitenkin piilossa käyttäjältä, ja niitä käsitellään rajapinnan funktioiden avulla. Olioiden tunnisteina toimivat kokonaisluvut. Suurin osa oliotyypeistä noudattaa samanlaisia käytäntöjä. Olennaisin poikkeus ovat shaderit, joista kerrotaan lisää myöhemmin tässä artikkelissa.

Aluksi oliolle täytyy varata tunniste. Tähän käytetään glGenTypes-funktiota, jossa Type on olion tyyppi. Funktiolla voi varata samalla kertaa useamman tunnisteen. Oliolla ei tässä vaiheessa ole vielä mitään tilaa.

Olio alustetaan, kun se sidotaan ensimmäisen kerran. Sidontafunktion nimi on muotoa glBindType. Funktio ottaa aina parametrina sidottavan olion tunnisteen. Tyypistä riippuen se saattaa ottaa myös sidontapisteen tai muita parametreja.

Sidottuna olevan olion tilaa voi muuttaa sen tyyppiin liittyvillä funktioilla. Nämä vaihtelevat rajusti olion tyypin mukaan. Useimmat oliot eivät ole käyttökelpoisia tai ainakaan hyödyllisiä heti luomisen jälkeen, vaan ne vaativat jonkinlaisen alustuksen.

Kun oliota ei enää tarvita, se tuhotaan glDeleteTypes-funktiolla. Vapautuksen suhteen on syytä olla huolellinen, sillä näytönohjaimella on rajallisesti muistia, ja vuotoja voi olla vaikea havaita.

Tunniste 0 on varattu tarkoittamaan olematonta oliota. Sitä käytetään voimassa olevan sidonnan purkamiseen. Joissain erikoistapauksissa 0 viittaa järjestelmän tarjoamaan olioon, kuten näytöllä näkyvään kuvapuskuriin.

Puskurit

Kuten edellisessä osassa kerroimme, OpenGL:lle syötettävä data täytyy ensin sijoittaa näytönohjaimen muistiin. Tähän käytetään puskureita. Ne ovat yleiskäyttöisiä muistialueita, joihin voi periaatteessa tallentaa mitä tahansa.

Puskurien sidontafunktio glBindBuffer ottaa parametrina puskurin lisäksi sidontapisteen. Käytössä on useita sidontapisteitä kommunikointiin OpenGL:n eri osien kanssa. Minkä tahansa puskurin voi sitoa mihin tahansa sidontapisteeseen, mutta implementaatio voi käyttää puskurin luomisen yhteydessä annettua sidontapistettä vihjeenä puskurin aiotusta käytöstä. Tällä voi olla merkitystä, mikäli näytönohjaimella on erilaisia muistialueita.

```
void glGenBuffers(GLsizei n, GLuint *buffers)
void glBindBuffer(GLenum target, GLuint
buffer)
void glDeleteBuffers(GLsizei n, const GLuint
*buffers)
```

```
Joitakin puskurien sidontapisteitä
GL_ARRAY_BUFFER: kulmapisteiden tiedot
GL_ELEMENT_ARRAY_BUFFER: indeksit
GL_PIXEL_UNPACK_BUFFER: tekstuuriin ladattava
pikselidata
GL_UNIFORM_BUFFER: shaderien uniform-muuttujat
```

Vasta luotu puskuri on vielä tyhjä. Se ei sisällä dataa, eikä sille ole edes varattu muistia. Sisältö määritellään glBufferData-funktiolla. Samalla funktiolla voi myös suorittaa pelkän muistin varauksen antamalla sille null-osoittimen. Tästä on hyötyä, jos varsinainen sisältö halutaan ladata puskuriin muulla tavoin, kuten esimerkiksi osittaisilla päivityksillä tai muualta OpenGL:stä lukemalla. Kutsuttaessa glBufferData-funktiota uudestaan samalle puskurille kaikki puskurissa ollut aiempi data hylätään.

```
void glBufferData(GLenum target, GLsizeiptr
size, const GLvoid *data, GLenum usage)
```

Muistinvarauksen yhteydessä voi antaa myös vihjeen siitä, kuinka usein sisältöä aiotaan päivittää, ja mihin suuntaan dataa siirretään. Kuten sidontapisteen tapauksessa, ajurit voivat käyttää tätä tietoa päättäessään puskurin sijoittamisesta muistiin. Epätarkan vihjeen antaminen ei estä puskurin käyttöä, mutta voi hidastaa sitä.

```
Puskurin käyttövihje on muotoa GL_PT_SS, jossa
PT kertoo päivitystiheyden ja SS siirtosuunnan
Päivitystiheys:
STATIC: vain kerran
DYNAMIC: silloin tällöin
STREAM: (lähes) jokaisen käytön yhteydessä
Siirtosuunta:
DRAW: käyttäjältä OpenGL:lle
READ: OpenGL:ltä käyttäjälle
COPY: OpenGL:n sisällä
```

Osaa puskurin sisällöstä voidaan muuttaa glBufferSubData-funktiolla. Puskurille täytyy tällöin olla etukäteen varattu muistialue, eikä sen kokoa voi osittaisen päivityksen yhteydessä muuttaa. Päivitettävä alue ei saa ylittää varatun muistialueen rajoja.

```
void glBufferSubData(GLenum target, GLintptr
offset, GLsizeiptr size, const GLvoid *data)
```

Kulmapistetaulukot

Yleisintä puskureissa säilytettävää tietoa ovat kulmapisteiden ominaisuudet. Kunkin ominaisuuden sijainti muistissa on määritetty erikseen. Ne voivat sijaita erillisillä muistialueilla, jopa eri puskureissa, tai lomittain sijoitettuna yhdellä muistialueella. Yleensä kannattaa käyttää lomitusta, koska tällöin saman kulmapisteen kaikki ominaisuudet ovat lähekkäin, mikä helpottaa niiden hallintaa ja parantaa välimuistin hyödyntämistä.

Kulmapisteiden ominaisuudet kootaan yhteen kulmapistetaulukoksi. Jokaista erilaista piirrettävää objektia varten tarvitaan oma kulmapistetaulukko.

```
void glGenVertexArrays(GLsizei n, GLuint
*arrays)
void glBindVertexArray(GLuint array)
void glDeleteVertexArrays(GLsizei n, const
GLuint *arrays)
```

Ominaisuustaulukon sijainti muistissa määritellään glVertexAttribPointer-funktiolla. Määrittely kohdistuu sidottuun kulmapistetaulukkoon. Haluttu puskuri on oltava sidottuna GL_ARRAY_BUFFER-sidontapisteeseen kutsuhetkellä. Puskuri tallennetaan osana sijaintia, joten sidonnan voi purkaa heti kutsun jälkeen. Sijaintia määriteltäessä on huomattava, että annettu osoitin tulkitaan puskurin alusta lukien. Null on siis aivan kelvollinen osoite ja tarkoittaa puskurin alkua.

Jos liukulukutyyppisen ominaisuuden syöte annetaan kokonaislukuina, voidaan arvot normalisoida. Tällöin kokonaislukutyypin suurin esitettävissä oleva arvo muunnetaan liukuluvuksi 1.0. Pienin esitettävissä oleva arvo muunnetaan liukuluvuksi 0.0 tai -1.0, riippuen siitä, onko kyseessä etumerkitön vai etumerkillinen kokonaisluku.

Lomitettua kulmapistetaulukkoa käytettäessä stride-parametrissa annetaan siirros kahden peräkkäisen kulmapisteen välillä.

```
void glVertexAttribPointer(GLuint index, GLint
size, GLenum type, GLboolean normalized,
GLsizei stride, const GLvoid *pointer)
```

OpenGL:lle on myös kerrottava, minkä numeroiset ominaisuudet tulee syöttää shaderille. Tähän käytetään funktioita glEnableVertexAttribArray ja glDisableVertexAttribArray. Kuten kaikki muutkin OpenGL:n tilat, myös käytössä olevat ominaisuudet säilyvät, ellei niitä erikseen muuteta. Yritys käyttää kulmapistettä määriteltyjen taulukoiden rajojen ulkopuolelta johtaa virheelliseen toimintaan tai ohjelman kaatumiseen. On siis syytä pitää huolta, että vain tarvittavat ominaisuudet ovat käytössä.

	Verteksin sijainti			Normaalivektori		
	X	Y	Z	X	Y	Z
float data[] = { 1 ➤	-1.00,	-1.00,	0.00,	3 ➤ 0.00,	0.00,	1.00,
2 ➤	1.00,	-1.00,	0.00,	0.00,	0.00,	1.00,
	-1.00,	1.00,	0.00,	0.00,	0.00,	1.00,
	1.00,	1.00,	0.00,	0.00,	0.00,	1.00 };

1. Ensimmäinen verteksi alkaa taulukon alusta
2. Verteksien välinen siirros on 6*sizeof(float) eli 24 tavua
3. Normaalin siirros verteksin alusta on 3*sideof(float) eli 12 tavua

Esimerkki lomitetusta verteksitaulukosta. Nämä verteksit muodostavat neliön, jonka normaali osoittaa ylöspäin

```
void glEnableVertexAttribArray(GLuint index)
void glDisableVertexAttribArray(GLuint index)
```

OpenGL-toteutuksen on tuettava vähintään 16 ominaisuutta, mutta se voi tukea useampaakin. Standardi ei ota kantaa niiden merkitykseen. On kuitenkin olemassa tiettyjä 3D-grafiikan peruskäsitteitä, joita kappaleiden piirtämiseen käytetään.

Sijainti on useimmissa tapauksissa pakollinen. Joissakin tehosteissa shaderi saattaa pystyä laskemaan sijainnin muiden ominaisuuksien perusteella. Yleensä käytetään kahden tai kolmen komponentin vektoreita. Sijaintiin viitataan usein englanniksi sanoilla vertex tai position.

Normaali on vektori, joka on kohtisuorassa pintaa vastaan. Interpoloinnin ansiosta sopivasti määritellyillä normaaleilla voi luoda illuusion tasaisen pyöreästä pinnasta. Normaalissa on lähes aina kolme komponenttia, ja sitä kuvaava englanninkielinen termi on normal.

Jos kappale on teksturoitu, tarvitaan tekstuurikoordinaatit. Monitekstuurointia käytettäessä tekstuurikoordinaatteja voi olla useita. Usein kuitenkin tekstuurit kuvaavat saman pinnan eri aspekteja ja käyttävät samoja koordinaatteja. Kaksiulotteisia kuvatekstuureja käytettäessä tekstuurikoordinaateissa on tavallisesti kaksi komponenttia. Englanninkielinen termi tekstuurikoordinaatille on texture coordinate, mutta usein se lyhennetään muotoon texcoord.

Jokainen kulmapiste on oma kokonaisuutensa, jolla on omat ominaisuudet. Koska ominaisuuksia ei ole mahdollista jakaa kulmapisteiden kesken, on usein tarpeen luoda useita kulmapisteitä osittain samoilla ominaisuuksilla. Esimerkiksi kuution jokaisen sivun neljällä kulmapisteellä on sama normaali, mutta eri sijainti. Jokaisessa kulmassa taas on kolme kulmapistettä, joilla on sama sijainti, mutta eri normaali.

Shaderit

Edellisessä numerossa esilteltiin myös lyhyesti shaderien toimintaperiaatteet. Nyt selvitämme tarkemmin, mitä näillä massiivista rinnakkaislaskentaa hyödyntävillä ohjelmanpätkillä voi tehdä.

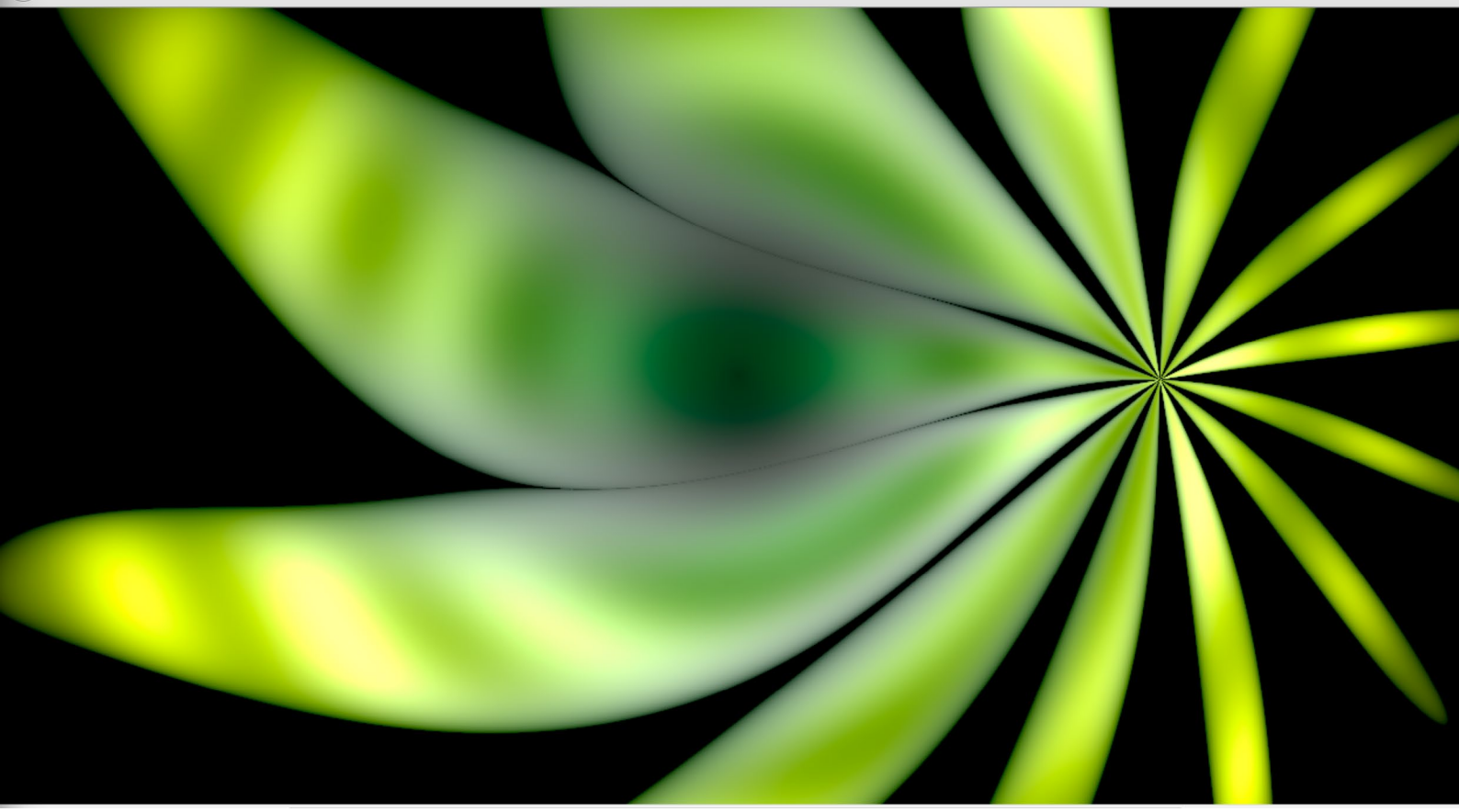
Shaderien suoritus jakautuu useaan vaiheeseen, joista kullakin on oma tehtävänsä. Kaikkien ohjelmointiin käytetään samaa C:tä muistuttavaa kieltä. Jotkin ominaisuudet ovat kuitenkin käytettävissä vain tietylle suoritusvaiheelle. Kunkin vaiheen tulokset annetaan syötteenä seuraavalle vaiheelle. Poikkeuksena ensimmäinen vaihe saa syötteensä kulmapistetaulukoista, ja viimeisen vaiheen tulokset määrittävät piirrettävän pikselin värin ja syvyyden.

Tarjolla on neljä perustietotyyppiä: bool, int, uint ja float. Nämä vastaavat karkeasti C++:n tietotyyppejä. Erillisiä tietotyyppejä eri kokoisille arvoille ei ole, vaan bool-tyyppiä lukuunottamatta kaikki perustietotyypit ovat 32-bittisiä.

Skalaarien lisäksi on tarjolla myös vektori- ja matriisityypit. Vektorissa voi olla kaksi, kolme tai neljä elementtiä. Matriisin rivien ja sarakkeiden määrä voi vastaavasti olla kaksi, kolme tai neljä.

```
Esimerkkejä shaderien tietotyypeistä:
bool    totuusarvo
int     etumerkillinen kokonaisluku
uint    etumerkitön kokonaisluku
float   liukuluku
vec4    4 alkion liukulukuvektori
ivec2   2 alkion kokonaislukuvektori
mat3    3x3-liukulukumatriisi
mat2x4  2 sarakkeen ja 4 rivin
liukulukumatriisi
```

GLSL ES:ssä tietotyypin jälkeen voi lisätä avainsanan lowp, mediump tai highp, joka määrittää matalan, keskitason tai korkean tarkkuuden. Matalammalla tarkkuudella laskeminen voi olla nopeampaa. Korkeaa tarkkuutta ei välttämättä ole tuettu kaikilla laitteilla. Myös normaali GLSL tunnistaa nämä avainsanat, mutta niillä ei ole vaikutusta.



Vektorin elementteihin voi viitata joko indeksoimalla tai jäsenmuuttujina. Jäsenille on kolme eri nimeä, jotka on nimetty avaruuskoordinaattien, tekstuurikoordinaattien ja värikomponenttien mukaan. Näistä voi käyttää sopivia sen mukaan, mitä vektori esittää. Ero on kuitenkin puhtaasti kosmeettinen eikä vaikuta shaderin toimintaan.

Aritmeettiset operaatiot toimivat pääosin kuten voi odottaa. Muutamia oikoteitäkin on tarjolla. Vektoriin tai matriisiin voi lisätä tai vähentää skalaarin, jolloin operaatio suoritetaan erikseen jokaiselle elementille. Kahden vektorin välinen kerto- tai jakolasku suoritetaan komponenteittain.

Vektoreilla on myös swizzlenä tunnettu erikoisoperaatio. Siinä vektorista valitaan kerralla useampi elementti, ja niistä muodostetaan uusi vektori. Komponenttien järjestys voi muuttua, ja sama komponentti voidaan valita useampia kertoja. Esimerkiksi v.xzyy vastaa samaa kuin vec4(v.x, v.z, v.y, v.y).

Teksturointia varten on olemassa joukko erityisiä näytteistäjätyyppejä (engl. sampler). Kukin tyyppi vastaa tietynlaista tekstuuria. Näytteistäjät ovat läpinäkymättömiä, ja niitä voi käyttää ainoastaan teksturointifunktioiden yhteydessä. Tutustumme niihin tarkemmin seuraavassa numerossa.

Lopuksi on vielä void-tyyppi, tai pikemminkin tyypin puute. Sen ainoa käyttötarkoitus on sellaisten funktioiden määrittely, jotka eivät palauta mitään.

Shaderin suoritus alkaa kunkin vaiheen main-funktiosta. Tämä funktio ei ota parametreja, eikä sillä ole paluuarvoa, joten sen tyyppi on void main(). Suoritus päättyy main-funktion loppuun.

Eri vaiheiden välinen rajapinta määritellään globaaleilla muuttujilla, jotka on esitelty in- ja out-avainsanoja käyttäen. Edellisen vaiheen tulosten ja seuraavan vaiheen syötteiden täytyy vastata toisiaan nimiltään ja tyypeiltään. Kullakin vaiheella on myös sisäänrakennettuja syöte- ja tulosmuuttujia kommunikointiin renderöintiputken muiden osien kanssa.

Kulmapisteshaderin on kirjoitettava verteksin sijainti gl_Position-muuttujaan. Tätä sijaintia käytetään primitiivien rasterointiin. Sen on oltava kuvaruudun koordinaatistossa eli kaikkien komponenttien on oltava -1:n ja 1:n välillä. Yleensä sijainti lasketaan malli-, näkymä- ja projektiomatriisien avulla (engl. model, view, projection).

Pikselishaderi saa syötteenä gl_FragCoord-muuttujassa pikselin interpoloidun sijainnin. Myös muut syötteet ovat interpoloituja. Shaderin on määriteltävä tulosmuuttuja, johon se kirjoittaa piirrettävän pikselin värin. Vaihtoehtoisesti tähän voi käyttää sisäänrakennettua gl_FragColor-muuttujaa. Nykyisin kuitenkin suositellaan muuttujan määrittelemistä itse. Halutessaan shaderi voi muuttaa pikselin syvyysarvoa kirjoittamalla uuden arvon gl_FragDepth-muuttujaan.

Kaikki shaderivaiheet voivat myös käyttää syötteitä, joiden arvo pysyy samana koko piirtokomennon ajan. Tällaiset muuttujat esitellään uniform-avainsanalla. Niiden arvot asetetaan isäntäohjelman puolelta ennen piirtokomennon suoritusta.

Shaderien käyttö

GLSL-ohjelmia ei voi käyttää sellaisenaan, vaan ainoastaan OpenGL-rajapinnan kautta. Niihin liittyviä olioita käsitellään hieman eri tavalla kuin muita. Luontifunktio on nimeltään glCreateType ja palauttaa luodun olion tunnisteen. Olio tuhotaan funktiolla glDeleteType. Oliota voi käsitellä tarvitsematta sitoa sitä ensin.

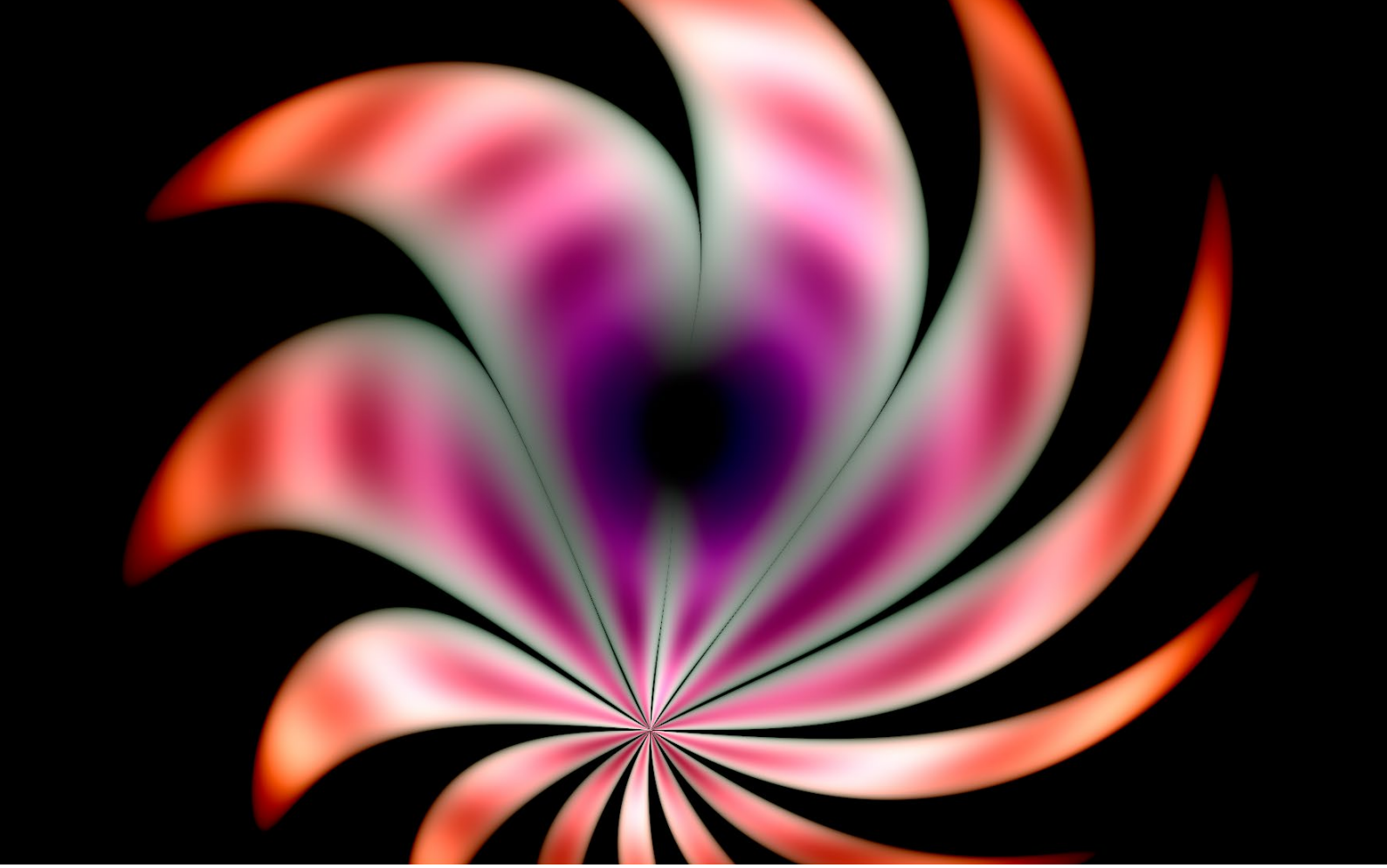
Shader-tyyppi kuvaa yksittäistä shaderin suoritusvaihetta. Luontifunktio ottaa parametrina haluttua suoritusvaihetta ilmaisevan vakion.

```
GLuint glCreateShader(GLenum type)
void glDeleteShader(GLuint shader)
```

Shaderin sisältö syötetään OpenGL:lle lähdekoodimuodossa glShaderSource-funktiolla. Se ottaa yhden tai useamman merkkijonon, jotka liitetään yhteen käännettäväksi lähdekoodiksi. Tämä mahdollistaa yhteisten osien sisällyttämisen

vec4-tietotyyppi muistuttaa tätä C:n unionia:

```
union vec4
{
  float array[4];
  struct  // Avaruuskoordinaatit
  {
    float x, y, z, w;
  };
  struct  // Tekstuurikoordinaatit
  {
    float s, t, p, q;
  };
  struct  // Väri
  {
    float r, g, b, a;
  };
};
```



erillisestä merkkijonosta C++:n otsaketiedostojen tapaan.

Grafiikkaprosessori ei kuitenkaan suorita lähdekoodia. Tätä varten ajureihin on upotettu kääntäjä, joka tuottaa käytettävälle prosessorille sopivan konekielikoodin. Kääntäjää voi kutsua lähdekoodin antamisen jälkeen glCompileShader-funktiolla.

```
void glShaderSource(GLuint shader, GLsizei
count, const GLchar **string, const GLint
*length)
void glCompileShader(GLuint shader)
```

Eri suoritusvaiheet on vielä yhdistettävä käyttöä varten kokonaiseksi ohjelmaksi. Tähän sopivan olion tyyppi on Program.

```
GLuint glCreateProgram()
void glDeleteProgram(GLuint program)
```

Shaderit liitetään ohjelmaan glAttachShader-funktiolla. Samaa suoritusvaihetta kuvaavia shadereita voi olla useita, mutta tasan yhdessä on oltava main-funktio. Sama shaderi voi myös olla useammassa ohjelmassa. Jotta ohjelma olisi käyttökelpoinen, siinä on oltava kulmapiste- ja pikselishaderi.

```
void glAttachShader(GLuint program, GLuint
shader)
```

Lopuksi ohjelma on linkitettävä glLinkProgram-funktiolla. Ohjelmaan liitettyjen shaderien on oltava käännettynä ennen tätä.

```
void glLinkProgram(GLuint program)
```

Kun kaikki on valmista, ohjelma voidaan ottaa käyttöön glUseProgram-funktiolla. Epätavallisesta nimestään huolimatta tämä toimii kuten muutkin sidontafunktiot.

```
void glUseProgram(GLuint program)
```

GLSL-ohjelman ulkoinen rajapinta muodostuu uniform-muuttujista ja kulmapisteshaderin syötteistä. Näihin viitataan isäntäohjelmassa niiden sijaintia kuvaavilla kokonaisluvuilla. Sijainnit saa nimen perusteella selville glGetAttribLocation- ja glGetUniformLocation-funktioilla. Jos kysyttyä muuttujaa ei ole määritelty shaderissa, funktiot palauttavat negatiivisen arvon.

```
GLint glGetAttribLocation(GLuint program,
const GLchar *name)
GLint glGetUniformLocation(GLuint program,
const GLchar *name)
```

Kulmapisteshaderin syötteet annetaan kulmapistetaulukoiden kautta, kuten edellä kerrottiin. Koska kulmapisteiden ominaisuudet on sidottu kulmapistetaulukossa tiettyihin sijainteihin, on käytännöllistä sitoa myös shaderin syötteet samoihin sijainteihin. Näin samaa kulmapistetaulukkoa voi käyttää useiden eri shaderien kanssa. Sidonta tehdään glBindAttribLocation-funktioilla. Ne tulevat voimaan seuraavan kerran ohjelmaa linkitettäessä.

```
void glBindAttribLocation(GLuint program,
GLuint index, const GLchar *name)
```

Uniform-muuttujien asettamiseen on joukko funktioita, omansa kullekin tyypille. Käytetyn funktion on vastattava asetettavan muuttujan tyyppiä, muutoin seurauksena on virhe. Poikkeuksia ovat totuusarvot ja niistä koostuvat vektorit, jotka voi asettaa minkä tahansa tyyppisellä funktiolla.

Useimmista muista shadereihin liittyvistä funktioista poiketen uniform-muuttujien asettaminen kohdistuu aktiiviseen ohjelmaan. Asetetut arvot ovat sidoksissa tähän ohjelmaan ja säilyvät, kunnes muuttujalle asetetaan uusi arvo.

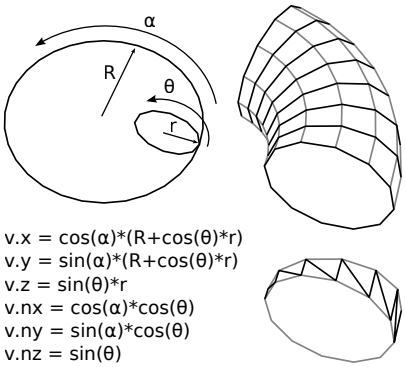
```
Joitakin uniform-muuttujien asettamiseen
käytettäviä funktioita:
void glUniform1f(GLint location, GLfloat v0)
void glUniform4f(GLint location, GLfloat v0,
GLfloat v1, GLfloat v2, GLfloat v3)
void glUniform4fv(GLint location, GLsizei
count, const GLfloat *value)
void UniformMatrix4fv(GLint location, GLsizei
count, GLboolean transpose, const GLfloat
*value)
```

Esimerkkiohjelma

Tämänkertaisessa esimerkkiohjelmassa loihdimme ruudulle pyörivän renkaan. Tätä varten lisäämme kulmapistettä kuvaavaan tietorakenteeseen kolmannen

koordinaatin ja normaalin. Kulmapistetaulukon täytämme ohjelmallisesti. Tarvitsemme myös indeksitaulukon, koska kulmapisteiden käyttäminen järjestyksessä ei enää riitä.

```
struct Vertex
{
  float x, y, z;
  float nx, ny, nz;
};
```



v.x = cos(α)*(R+cos(θ)*r)
v.y = sin(α)*(R+cos(θ)*r)
v.z = sin(θ)*r
v.nx = cos(α)*cos(θ)
v.ny = sin(α)*cos(θ)
v.nz = sin(θ)

Kullakin kulmapisteellä on nyt kaksi ominaisuutta. Turhan toiston välttämiseksi teemme apufunktion ominaisuustaulukoiden asettamiseksi. C++:n templateilla ja jäsenosoittimilla saamme aikaan kompaktin syntaksin funktion käytölle.

```
template<typename T>
void set_attrib_array(unsigned index, unsigned
size, float T::*member)
{
  T dummy;
  const char *offset_base = reinterpret_
cast<const char *>(&dummy);
  const char *const offset = reinterpret_
cast<const char *>(&(dummy.*member))-offset_
base;
  glVertexAttribPointer(index, size, GL_FLOAT,
false, sizeof(T), offset);
  glEnableVertexAttribArray(index);
}

set_attrib_array(POSITION, 3, &Vertex::x);
set_attrib_array(NORMAL, 3, &Vertex::nx);
```

Koska haluamme piirtää kolmiulotteisen kappaleen, täytyy syvyyspuskuri ottaa käyttöön. Sen avulla pikseli jätetään piirtämättä, jos samaan kohtaan on jo piirretty lähempänä oleva pikseli.

```
glDepthFunc(GL_LEQUAL);
glEnable(GL_DEPTH_TEST);
```

Myös shaderit vaativat muutoksia. Kulmapisteshaderin on tehtävä tarvittavat koordinaattimuunnokset. Malli- ja näkymämatriisit on tavanomaista yhdistää yhdeksi matriisiksi, joka määrittää muunnoksen objektin koordinaateista näkökentän koordinaatteihin.

```
uniform mat4 modelview;
uniform mat4 projection;
in vec4 position;
in vec3 normal;
out float intensity;

void main()
{
  gl_Position = projection*modelview*position;
  intensity = (mat3(modelview)*normal).z;
}
```

Kulmapisteshaderi laskee myös valaistuksen voimakkuuden kullekin verteksille ja välittää sen pikselishaderille. Pikselishaderissa interpoloitu valaistus yhdistetään pinnan väriin, jotta saadaan lopullinen piirrettävän pikselin väri.

```
uniform vec4 color;
in float intensity;
out vec4 out_color;

void main()
{
  out_color = vec4(color.rgb*intensity,
color.a);
}
```

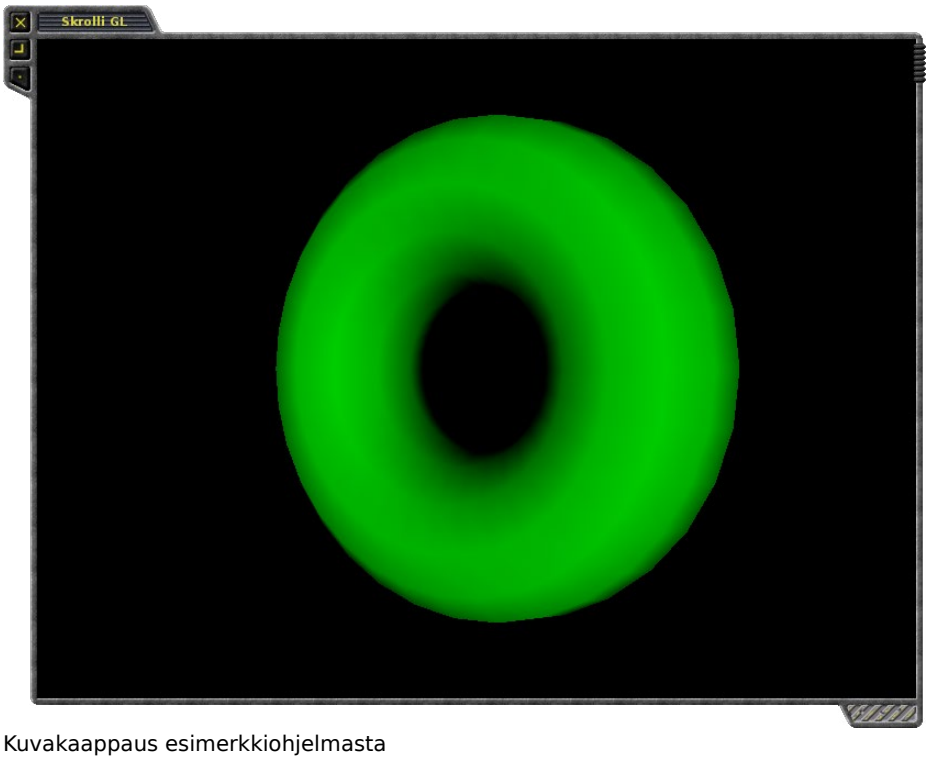
Pyöritystä varten annamme modelview-matriisille uuden arvon ennen jokaista ruudunpäivitystä. Koska katselupiste on origossa, täytyy kappaletta siirtää kauemmas, jotta se mahtuisi ruudulle. Pyöritys tehdään matriisin kolmella ensimmäisellä sarakkeella ja siirto neljännellä.

```
float modelview[16] =
⎡ cosα   0.0f   -sinα   0.0f ⎤
| 0.0f   1.0f    0.0f   0.0f |
| sinα   0.0f    cosα   -8.0f|
⎣ 0.0f   0.0f    0.0f   1.0f ⎦;
glUniformMatrix4fv(modelview_loc, 1, false,
modelview);
```

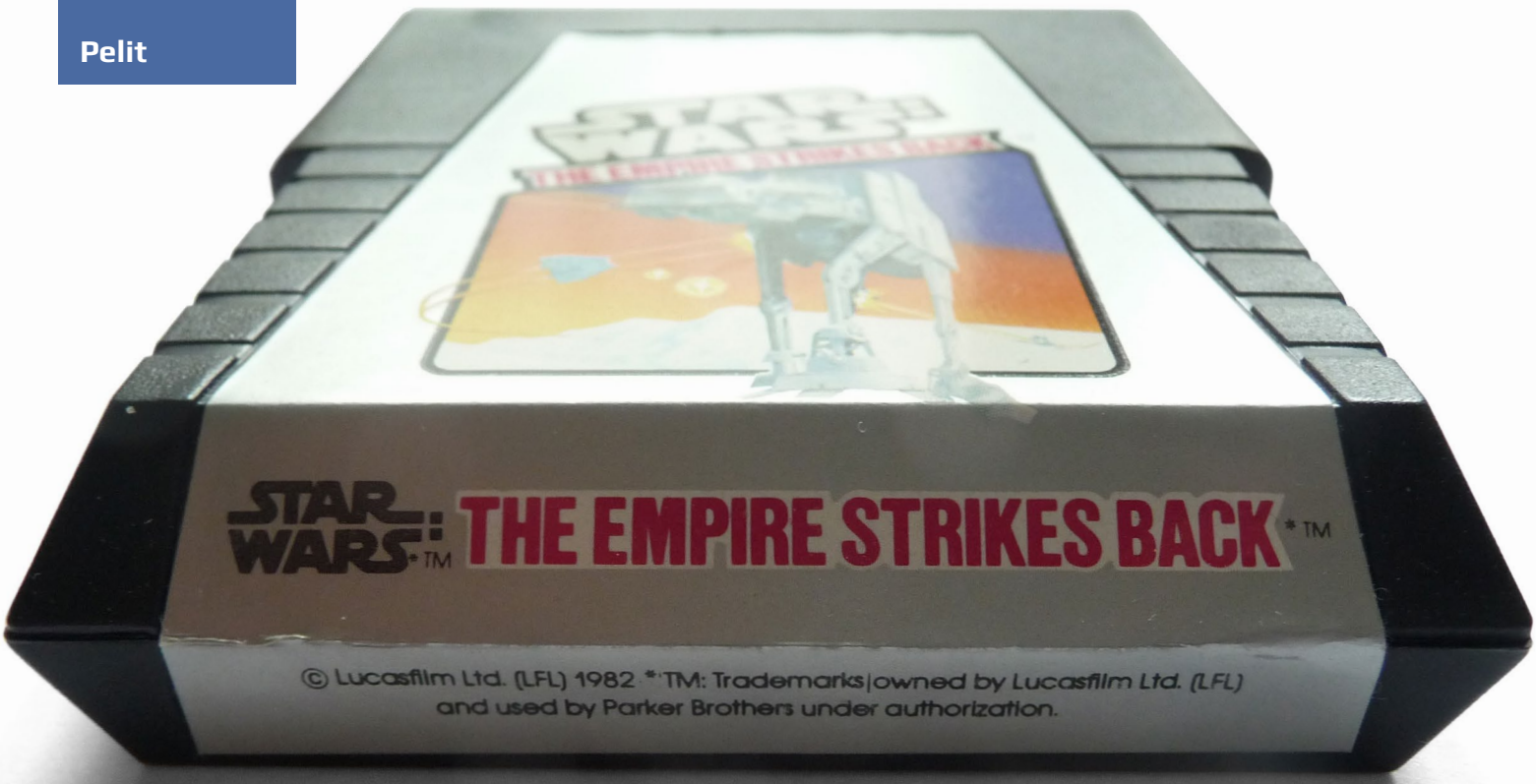
Esimerkkiohjelman täydellinen lähdekoodi löytyy Skrollin nettisivuilta osoitteesta http://skrolli.fi/2013/ .

Tehtäviä ja haasteita lukijalle:

- Lähellä ohjelman alkua on joitakin vakioarvoja, jotka määrittelevät renkaan koon ja käytettävien tasopintojen määrän. Kokeile erilaisia arvoja ja katso, mitä tapahtuu.
- Pystytkö muuttamaan kappaleen kokonaan eri muotoiseksi? Onnistuuko pallo, sylinteri tai kuutio?
- Shaderille annetaan piirrettävän kappaleen väri uniform-muuttujana. Saatko kappaleen vaihtamaan väriä pyörityksen aikana?
- Osaatko lisätä kulmapisteille värin ja luoda monivärisen kappaleen?
- Kokeile muuttaa modelview-matriisin sisältöä siten, että kappale on ruudulla eri kohdassa tai pyörii eri suuntaan. Saatko sen liikkumaan pyörimisen sijaan, tai tekemään molempia samanaikaisesti?
- Saatat huomata pinnan kirkkauden vaihtelussa epätasaisuutta, joka johtuu verteksikohtaisen valaistuksen käytöstä. Osaatko toteuttaa pikselikohtaisen valaistuksen?



Kuvakaappaus esimerkkiohjelmasta



Kaukaisessa galaksissa, vuonna 1982...

Jediritareiden universumista on vuosien varrella siinnyt pitkä ketju tähteinvälisiä pelejä. Vilkaisemme ensimmäisten syntyhistoriaa Rex Bradfordin seurassa.

Teksti: Juho Pietarinen

Kuvat: Juho Pietarinen, Toni Kuokkanen

On mahdotonta kuvitella pelien nykymaailmaa ilman Star Wars –universumiin sijoittuvia karkeloita. Niitä kun löytyy nykyään joka makuun — on Legoja, klooneja ja MMO:ta. Harva tulee ajatelleeksi, että jostain pelistä se tähtien seassa poukkoilukin lähti liikkeelle.

Star Wars -pelien ensiversio on Atari 2600:n Empire Strikes Back. Se löytyy lautapeleistään tunnetun Parker Brothersin pieneltä peliosastolta vuodelta 1982.

Siellä Atari 2600:n tuskaisen alitehoiselle raudalle onnistuttiin siirtämään elokuva, joka oli juuri voittanut Oscarit mahtavista erikoistehoisteistaan. Siirto ei kuitenkaan sujunut aivan ongelmitta.

Parker Brothers

Star Wars -peleistä ensimmäistä ohjelmoimaan valikoitui märkäkorva Rex Bradford, joka oli siihen asti lähinnä puurtanut Parker Brothersin ykköstuotteen, Monopolin, elektronisen version kimpussa. Empire Strikes Back oli sekä Rexin että koko Parker Brothersin ensimmäinen konsolipeli.

Rex muistelee vanhaa työpaikkaansa lämmöllä. "Elektroniikkaporukalla oli hallussaan kokonainen kerros pääkonttorista", kertoo pelivelho. "Se oli perinteinen jenkkifirma hienoine pukuineen, ja me olimme siinä joukossa kummajaisia. Aina kun emoyhtiön pomot tulivat vierailulle, meidän piti siivota pöydät kiiltäviksi ja vetäistä päälle valkoiset labratakit — oikeasti!"

Atari 2600:n siirtyminen pelimarkkinoiden valloittajaksi ei ollut aivan yksinkertaista. Atari varjeli markkinoita hallitsevan konsolinsa grafiikkajärjestelmän saloja kuin orava ainoaa käpyään. Säilyttämällä sen tekniset yksityiskohdat mysteerinä Atari toivoi pitävänsä konsolinsa pelimarkkinat yksinoikeutenaan.

Tuo yksinvaltius oli kuitenkin jo alkanut rapistumaan. Activision, jonka oli perustanut neljä Atarilta livahtanutta ykkösrivin pelintekijää, oli ensimmäisenä ulkopuolisena alkanut tuottamaan pelejä Atari 2600:lle jo pari vuotta aiemmin. Parker Brothers oli heti seuraavana haaskalla.

Lepertelyä konekielellä

Rexin ensimmäisenä tehtävänä oli opetella konsolin tuskaisen alitehoisen 6507-prosessorin konekieli ja kirjoittaa sille purkuohjelma. Purkuohjelman avulla tiimi voisi tutkia mistä Atarin pelit oli tehty, ja ottaa niistä ottaa mallia omaa peliä luodessaan.

Samalla Jim McGinnis Parker Brothersin pomoportaasta ja ohjelmointitiimin pääjehu Mark Lesser ottivat tehtäväkseen selvittää valokuvien avulla, mitä salaisen grafiikkajärjestelmän helmojen alta löytyi. Ne valokuvat ovatkin sitten osa bittihistorian komediakultaa.

"Parker pestasi läheisen firman viilaamaan kerroksia grafiikkapiirin päältä ja valokuvaamaan niitä, jotta Mark ja Jim pystyisivät tutkimaan kuvia ja takaisinmallintamaan grafiikkapiirin niiden avulla. Ensimmäiset valokuvat olivat todella suttuisia, ja firma epäili, että läheisen tien rekkaliikenteen aiheuttama tärinä heilutti kameraa."

Firma pähkäili ongelmaan ratkaisua ja päätyi lopulta rakentamaan erillisen elohopeaastian, jolla tasapainottaa kameraa. Tämä ei kuitenkaan tuonut parannusta: supertarkat kuvat olivat yhä suttuisia.

"Lopulta joku huomasi, että kameran linssin eteen oli jäänyt pala suojateippiä koko ajaksi!", Rex nauraa.

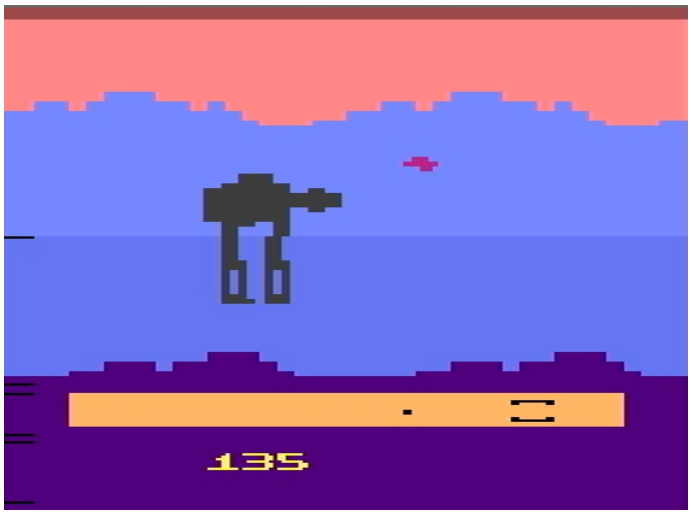
Star Wars – Empire Strikes Back

Julkaisija: Parker Brothers

Kehittäjä: Rex Bradford ja Sam Kjellman

Julkaisuvuosi: 1982

Genre: Sivuttain vierivä ammuskelu



Kohtalon hetket käynnissä Hothissa.

Rex Bradford, SOLMIO-hävittäjä

Star Wars –pelin tekeminen oli Rexille unelmien täyttymys.

"Olin nähnyt tuolloin Empire Strikes Backin jo varmaan kymmenen kertaa teatterissa", mies muistelee. Hän ryhtyikin peliä tekemään kuin "tykin suusta ammuttuna".

Toisin kuin Atarilla, jossa tietyn pelin ohjelmoija käytännössä sulkeutui yksin luolaansa kuukausiksi ja ilmestyi sieltä lopulta hienhajun ja uuden pelin kera, oli Parker Brothers osa Activisionin aloittamaa uutta aaltoa. Siinä peleistä vastasivat pienet tiimit. Ensimmäistä Star Warsia suunnittelemaan ryhtyi Sam Kjellman ja ohjelmoinnista vastasi Rex. Myös Mark Lesser liittyi porukkaan. Mark oli jo lunastanut paikkansa pelialan historiassa, sillä mies oli tehnyt alusta loppuun Mattel Auto Racen, kaikkien aikojen ensimmäisen kannettavan elektroniikkapelin. Oli itse asiassa Markin ansiota, että koko Star Wars -peliä oli ylipäätään ryhdytty tekemään.

Tarina kertoo, että eräänä hämyisenä iltana Parker Brotherin toimistolla Mark oli muiden puuhiensa lomassa taikonut toimiston Atari 2600:n ruudulle Darth Vaderin näköiskuvan. Parker Brothersin markkinointipomo Bill Bracy sattui huomaamaan sen ja vaikuttui näkemästään. Upeasta näköiskuvasta löytyi sekä värit että varjostukset, minkä piti olla Atari 2600:lla lähes mahdotonta. Tässä konsolissahan on potentiaalia, päätteli Bill, ja tiedusteli Markilta miten kuvan saisi siirrettyä peliin. Ei mitenkään, vastasi Mark. Kuva vei koko sen neljän kilotavun tallennustilan, mitä Atari 2600:n pelimoduulilta löytyi.

Star Wars -pelin aihe, elokuvasta tuttu Hothin taistelu, oli valikoitunut jo ennen Rexin liittymistä projektiin. Valinta osui nappiin. Pelaaja ohjaa pientä lumikiitäjää massiivisten kävelijöiden vyöryessä Imperiumin mahdin lailla kohti Kapinaliiton Echo-tukikohtaa. Elokuvalle uskollisesti hyökkäystä ei ole edes mahdollista pysäyttää.

Pelillä ei ole päätöstä. Ei lopputaistelua Darth Vaderia vastaan, jonka yhteydessä voisi pistää riidan ja käden poikki. Ilotikun vatkaajan kohtalona oli käydä väsyttävän rankkaa viivytystaistelua, jonka mahdollinen voitto laskettaisiin vain pisteillä.

Teknistä iloittelua

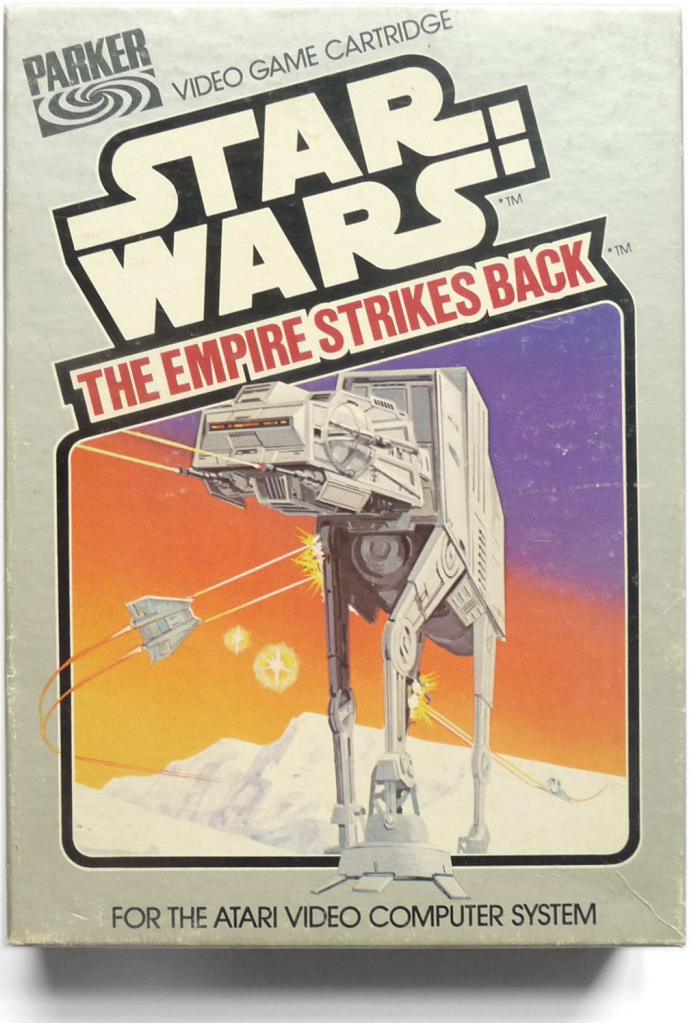
Rex kierteli vähätehoisen konsolin rajoja ovelasti, käsittelemällä ruudun eri osa-alueita itsenäisinä osina. Massiivisten kävelijöiden spritet jaettiin kahtia. Niiden runko sijoitettiin ruudun yläosaan ja suurennettiin nelinkertaiseksi. Jalat taas asettuivat ruudun alaosaan ja suurennettiin kaksinkertaisiksi. Molempien leveys oli kahdeksan pikseliä, mutta yläosa päätyi isomman suurennoksen vuoksi näyttämään kulmikkaammalta. Kävelijöiden spriten jakaminen osiin mahdollisti samalla myös niiden erilaisen käsittelyn. Rungosta tehtiin kiinteä, siihen ei pelaajan auttanut törmätä tai matka tyssäsi. Jalat taas olivat silkkaa ilmaa, joten kiiturilla pystyi luikahtamaan niiden "välistä" kuin elokuvassa konsanaan. Ohjusten räjäytykset Rex toteutti viekkaasti suurentamalla ohjuksen osumishetkellä sen spriteä nopeasti kahdeksankertaiseksi ja heiluttamalla sijaintipaikkaa.

Aina välillä kävelijöiden eteen tai taakse ilmestyi pelaajaa seireenin lailla houkutteleva vilkkuva pikseli, johon osumalla sai kävelijän kumoon yhdellä laukauksella. Normaali tykitys rasitti rannetta 48 osuman verran. Jos taas onnistui väistelemään kävelijöiden tykkien ohjuksia kahden minuutin ajan, luritteli konsoli ulos Star Wars -teeman ja pelaajan kiitäjä oli hetken Voiman valtaamana vahingoittumaton.

Nämä lisäykset toivat peliin taktisen elementin. Pelaaja pystyi joko sarjatulittamaan kävelijät ketoon ensimmäisestä viimeiseen, tai vain hidastamaan ensimmäistä. Tämä taktiikka sai koko jonon matelemaan kuin maatuskat kaupan kassalla, ja soi pelaajalle mahdollisuuden snaipata ärtyneet kävelijät rauhassa yksi kerrallaan.

Peli oli myös ensimmäinen, joka esitteli parallaksiskrollauksen kotikonsoleille. Siinä kuvan tausta liikkuu hieman etualaa hitaammin, ja antaa toiminnalle kolmiulotteisen vaikutelman.

"Se oli itse asiassa aika helppoa toteuttaa", toteaa Rex osallisuudestaan yhteen videopelien merkittävimmistä kehitysaskeleista. Parallaksiskrollaus esiintyi ensimmäisen kerran videopelissä pelihallien Moon Patrolissa vuonna 1982, samana vuonna kuin Empire Strikes Back oli valmis kotiyleisön ihmeteltäväksi. Myöhempiin konsoleihin kyky parallaksiskrollauk-



seen rakennettiin suoraan rautaan.

Tähtipölyn tiivistämistä

Empire Strikes Back valtasi etusivun Parker Brothersin ensimmäisestä videopelilehtisestä ja jätti varjoonsa jopa varmaksi menestyjäksi povatun Froggerin, jonka Atari-käännökseen oli ostettu oikeudet Konamilta. Samalla Parker Brothersin markkinamiehet intoutuivat vaatimaan peliin uusia ominaisuuksia, kuten kävelijöiden köysitystä ja räjähdysanimaatioita. Rex paljastaa livistäneensä niiden toteuttamisesta konsolin kyvyttömyyteen vedoten.

Tiimi pakersi pelin parissa yli viisi kuukautta. "Ohjelmointi 2600:lle oli todella hauskaa, mutta muistutti paljon enemmän sudokun täyttämistä kuin varsinaista ohjelmointia", Rex sanoo. Konsolin graafisia rekistereitä piti muokata reaaliajassa samaan tahtiin, kuin television kuvaputken elektronisuihku piirsi kuvaa ylhäältä alas. 6507-prosessorin käskykanta on kiinteän pituinen käskyjaksojensa osalta, joten kikkana oli keksiä kuinka uudelleenohjelmoida ruutua rivi kerrallaan, lukea joystickin liikkeet ja suorittaa kaikki muut laskut ruutupäivitysten välissä. Se sekunnin murtoosa, mitä television elektronisuihkulla kesti siirtyä alhaalta ylös ja aloittaa uuden kuvan piirtäminen, tunnettiin siunattuna hetkenä nimeltä vertical blanking period. Siihen pieneen hetkeen Rex änki siis kaiken muun kuin grafiikat. Niin musiikit, ohjausliikkeet kuin pelimekaniikankin. "Se oli todella hullunkurista!", Rex nauraa.

Ohjelmointitiimin Ed Temple päätyi tekemään jokaisesta käskystä oman magneettinsa ja asetteli niitä isolle taululle erilaisiin järjestyksiin. Rexin tarinat paljastavat, että mikään osa peliä ei vielä tuohon aikaan ollut itsestäänselvyys. "Muistan, että yhdessä vaiheessa peli alkoi tuntumaan varsin hauskalta, mutta muiden mielestä siitä yhä puuttui jotain", Rex kertoo. "Joten eräänä päivä pistin siihen asti mykkään peliin muutaman äänitehosteen ja vastaus oli: 'Mitä ihmettä sinä teit?! Peli on nyt mahtava!'"

Klassikon kopiomallit

Empire Strikes Back päätyi lopulta muistuttamaan markkinat kaksi vuotta aiemmin vallannutta kolikkopeli Defenderiä melko vahvasti. "En kopioinut sitä tietoisesti", Rex selvittää, vaikka myöntää olleensa pelistä tietoinen. Defender oli Williams Electronicsin massiivinen menestys ja myi yli 55 000 pelikabinettia. Siinä pelaaja rullaa aluksellaan sivusuunnassa, ampuen kaikkea mahdollista. Kuulostaako tutulta? Ehkä Rexin tiimi ei suoraan kopioinut peliä, mutta paljon samaa niissä on. Annettakoon se kuitenkin anteeksi, vastaavaa kaavaa kun noudatti aika moni muukin menestyspeli. "Taistelu kävelijöitä vastaan oli luonnollinen valinta ja sivuttain vieriminen väistämätöntä", Rex selventää. "Pelin kamera ja lumikiitäjän fysiikka rakentuivat perusversion jälkeen hiljalleen peliä muokaten ja säätäen, kunnes kaikki palat tuntuivat olevan oikeilla paikoillaan."

Nykyään Rex puuhaa yhä pelien parissa ja ylläpitää historiallisia salamurhia käsittelevää sivustoa nimeltään History Matters. Mies muistelee yhä erityisellä lämmöllä tuota ensimmäistä peliään.

Äärimmäisten rajojen viehätys

Atari 2600 on tekniikaltaan äärimmäinen laite, jolla monet yksinkertaisetkin asiat hipovat mahdottomuuden rajoja. Se on malliesimerkki laitteesta, jonka estetiikka on vahvasti sidoksissa tekniikkaan.

Teksti: Ville-Matias Heikkilä

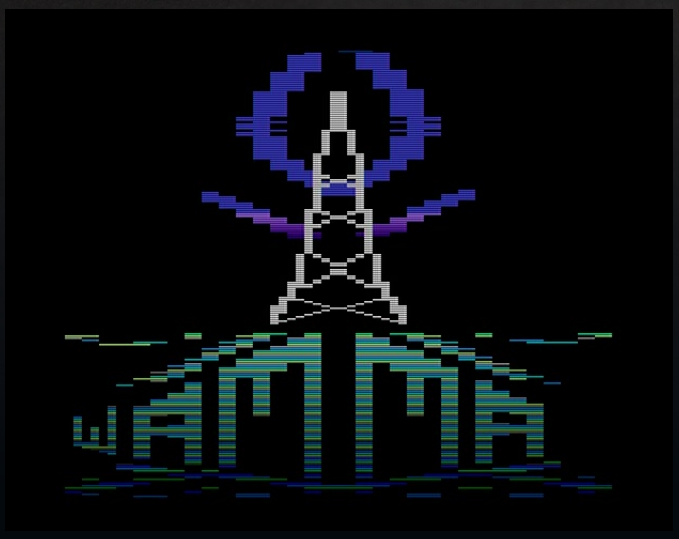
Kuvat: Waltteri Lahti, Toni Kuokkanen, Sven Oliver Moll

Atarin vuonna 1977 julkaiseman Video Computer Systemin, Atari 2600:n, sisällä on kolme mikropiiriä: 6502-yhteensopiva 6507-suoritin, video- ja äänipiiri TIA ja I/O-apupiiri 6532. Viimeksimainitun sisällä on myös 128 tavua RAM-muistia — kaikki, mitä koko koneesta löytyy. ROM-moduulille mahtuu 4096 tavua kiinteää ohjelmaa — tai enemmänkin, mikäli moduuliin rakennetaan sopiva sivutusmekanismi. Valtaosa peleistä on kuitenkin neljän kilotavun kokoisia.

2600:n merkittävin pullonkaula on RAM-muistin määrä. 128 tavun on riitettävä kaikille ohjelman muuttujille. Mistään näyttömuistin tapaisista isoista puskureista ei kannata edes haaveilla. Ainoa näyttömuistin kaltainen on TIA-piirin rekisteripankki, jonka bittien asennot kuvaavat, kuinka kuvan yksittäinen vaakajuova muodostetaan. Piiriä voisikin luonnehtia "1D-näytönohjaimeksi". Jotta näytölle saa mitään järkevää, on ohjelman seurattava näytönvirkistyksen etenemistä juova juovalta ja muutettava TIA-rekisterien tilaa sen tahtiin.

TIA-piirin ominaisuudet on arvatenkin optimoitu peligrafiikalle. Piiri kykenee tuottamaan kullekin juovalle taustakuvion ja sen päälle viisi liikutettavaa irtokuviota, spriteä. Taustakuvio on hyvin karkea, 40 pikseliä koko näytön leveydelle. Spriteistä kaksi on 8-pikselisiä bittikarttoja, ja ne saa näkymään samalla juovalla myös kahtena tai kolmena. Kolme muuta spriteä ovat pelkkiä tasavärisiä palikoita. Kunkin spriten leveyden voi valita muutamasta vaihtoehdosta. Taustakuviolle voi valita kaksi väriä ja kullekin spritelle yhden värin peräti 121-värisestä kokonaispaletista.

Äänipuolella Atarissa on



2000-luvun demotuotantoa: Wamma-ryhmän Gehirn

kaksi synteesikanavaa, joiden bitinpyöritykset ajastetaan 5-bittisillä laskureilla. Mahdollisia sävelkorkeuksia on siis hyvin vähän, ja melko harvojen välille saa puhtaita intervalleja. Laitteen tekniikka vaikuttaa vahvasti myös siihen, millaisia harmonioita ja melodioita sillä on mielekästä toteuttaa.

Mitä rajojen puitteissa syntyy?

Mediatutkijat Nick Montfort ja Ian Bogost käsittelevät 2600:n estetiikkaa kirjassaan "Racing the Beam". Mitä laitteelle sitten on tehtykin, tekniset rajat ovat tulleet hyvin äkkiä vastaan. Niihin sopeutuminen onkin synnyttänyt aivan oman estetiikkansa.

Vierekkäisyys on Atarilla vaikeampaa kuin päällekkäisyys, ja rinnakkain näkyvät hahmot on aina mietittävä tarkoin. Space Invadersin hyökkäyslaivueen leveys on kuusi ötökkää — kaksi triplattua bittikarttaspriteä. Video Chess -shakkipeli joutuu paitsi miettimään siirtonsa hyvin pienellä työmuistilla näyttö sammutettuna, myös piirtämään pelilaudan lomitettuna, sillä kahdeksaa nappulaa ei saa näkyviin samalle juovalle.

Puhdas 2D-grafiikka on Atarilla hyvin karkeaa verrattuna useimpiin kasibittisiin kotimikroihin, mutta erilaiset juovittaiset rasteritemput onnistuvat monesti paremmin. Tämä näkyy esimerkiksi Pole Position- ja Battlezone-pelien perspektiivitoteutuksissa. Battlezone myös hyödyntää koneen laajaa väripalettia kauniisti taivaan ja maan väriliu'uissa, mutta vihollistankkien esittäminen karkeina bittikarttoina palauttaa TIA:n rajoitteet takaisin mieleen.

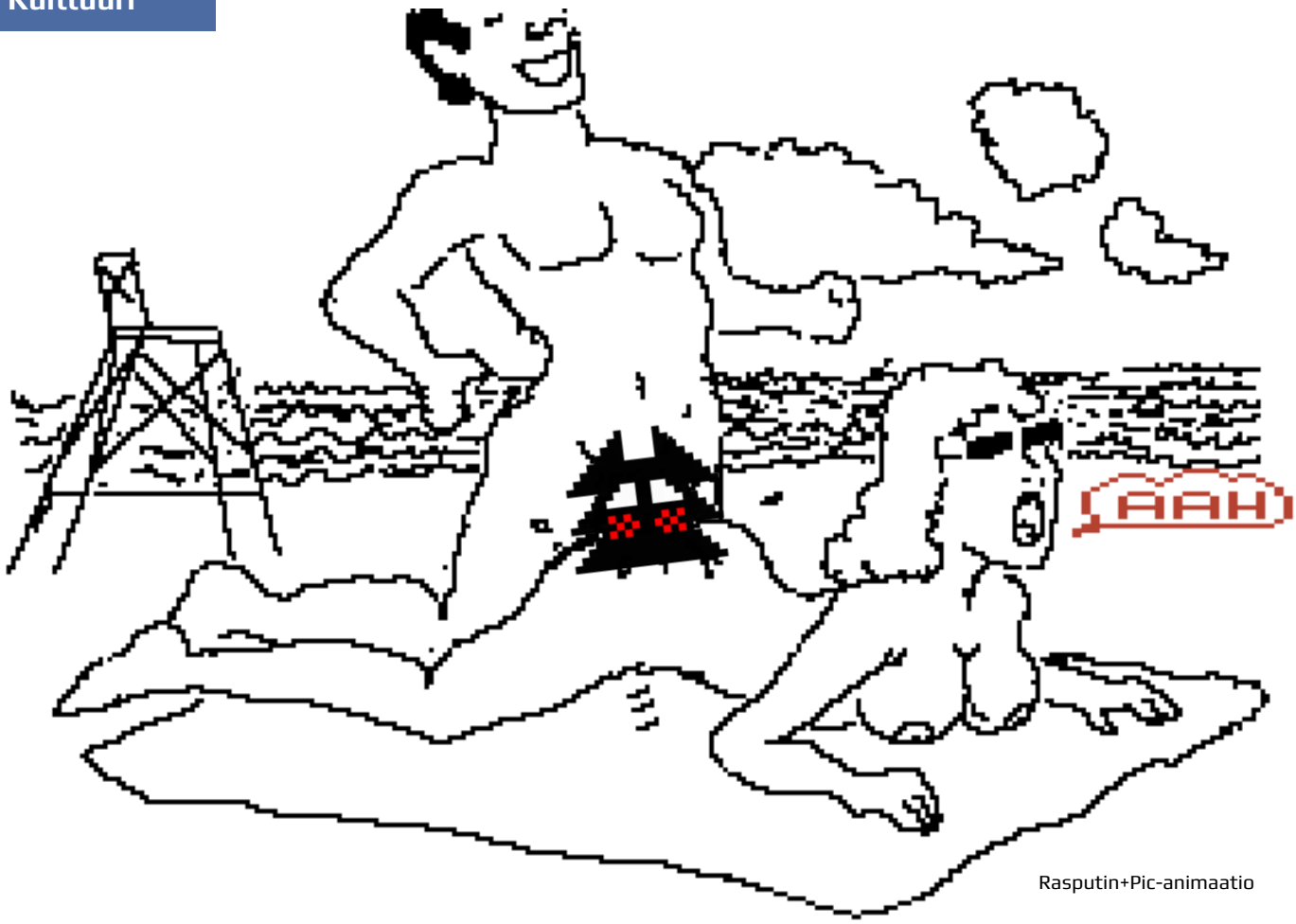
Kotipolttajat työn touhussa

Homebrew-kehittäjien keskuudessa 2600 on suosittu alusta. Suosiota selittää paljolti koneen historia, mutta monet varmasti motivoituvat myös sen teknisistä puitteista, jotka synnyttävät erikoisia ohjelmointihaasteita ja omalaatuista rujoa estetiikkaa.

Harrasteohjelmoijat ovat keskittyneet lähinnä peleihin, mutta 2000-luvulla myös muutama demoryhmä löysi koneen. Laitteiston rajoja pusketaan jatkuvasti eteenpäin. Uudet pelit vierittävät näyttöä moneen suuntaan, musiikki on moniäänistä ja kuvissa on yllättävän paljon värejä. Demoissa näkyy monia isommista koneista tuttuja perspektiivitemppuja, kuvanvenyttelyjä ja tietenkin plasmaefektejä. Useimmat uudetkin ohjelmat kunnioittavat perinteisiä rajoja, mutta esimerkiksi Boulder Dash -käännös joutuu huijaamaan hieman. Dynaamisesti muuttuva pelikenttä ei kerta kaikkiaan mahdu kerralla 128 tavun muistiin, joten pelimoduulissa on siksi oma RAM-laajennos.

Atari 2600 -kehitystä voi hyvin kokeilla myös ilman aitoa laitetta. Emulaattorin kaveriksi tarvitaan lähinnä 6502-ristiinkääntäjä ja TIA-piirin dokumentaatio. Konekieliohjelmointia karsastaville on tarjolla myös ilmainen BASIC-kääntäjä Batari Basic, jonka käyttöoppaassa käydään läpi konsolin toiminta peruskäsitteistä alkaen. Omat tuotokset on perinteisesti siirretty aitoon rautaan EPROM-piireillä, mutta nykyisin on tarjolla myös esimerkiksi Harmony Cartridge, joka käyttää massamuistina SD-korttia.

Battlezone-käännös vuodelta 1983



Rasputin+Pic-animaatio

Tikku-ukoista pikselitisseihin:
digitaalisen erotiikan varhaisvuodet

Tekniikka valjastetaan eroottisen materiaalin tarpeisiin heti ilmestyessään. Bittimuotoinen porno levisi jo ennen internetin läpimurtoa.

Teksti: Mikko Heinonen, Markku Reunanen, Petri Lankoski

Vaikka aiheesta puhutaan mieluiten sivulauseissa ja kuiskaten, erotiikka ja tekninen kehitys ovat kietoutuneet tiukasti yhteen. Varsin pian keksimisensä jälkeen painokone suolsi tuhmia kuvia, kameralla kuvattiin alastomuutta, filmikameralla taas eroottisia elokuvanpätkiä. Pornovideoiden saatavuuden sanotaan olleen eräs keinoista, joilla VHS-videoformaatti voitti taistelun kotitalouksien suosiosta. Eroottisen materiaalin tuottajat ovat aina löytäneet uudet mediat, eikä tietotekniikka ollut mikään poikkeus.

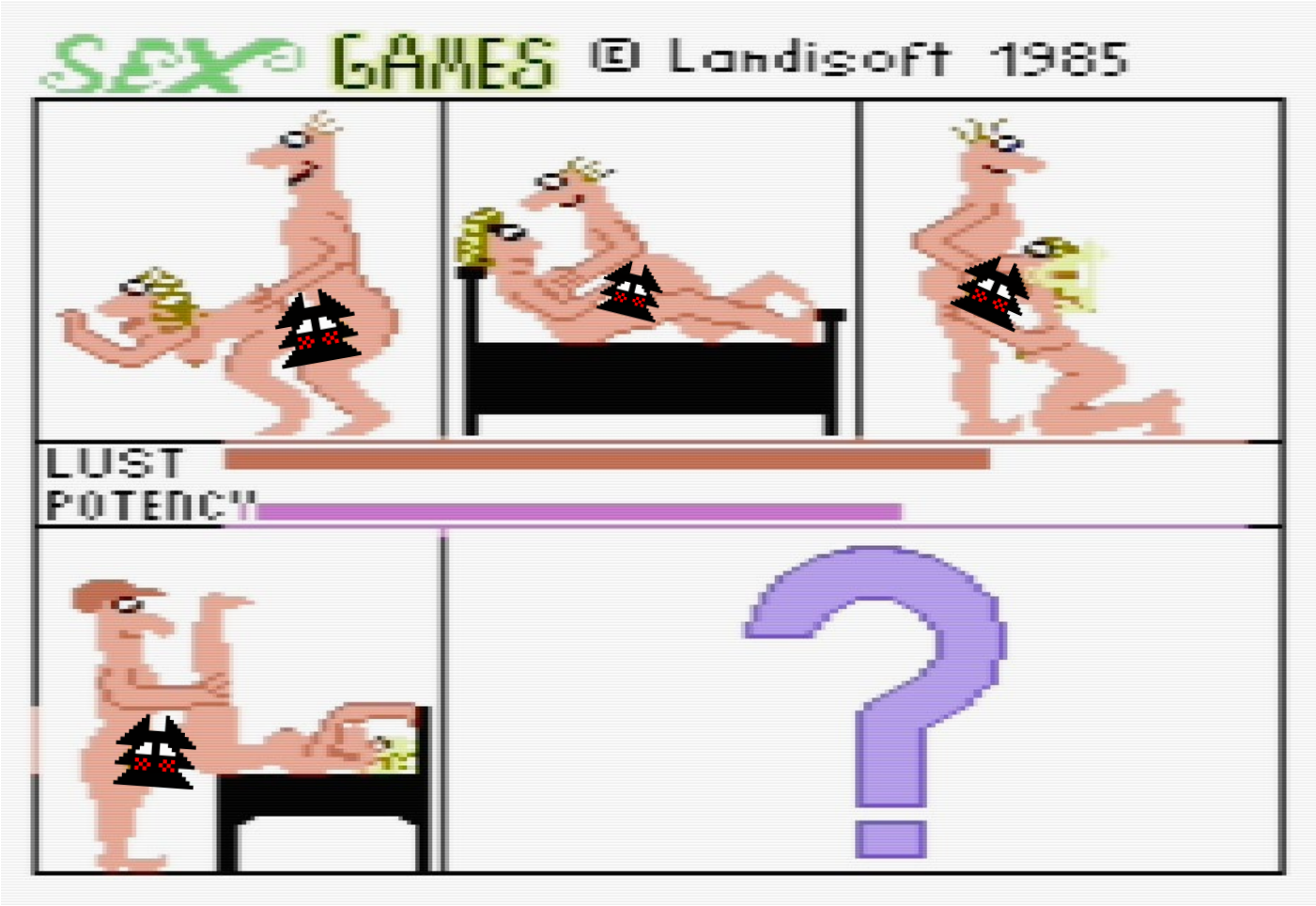
Ensimmäinen kaupallinen erotiikkapeli oli tiettävästi Sierra On-Linen Softporn Adventure vuodelta 1981. Tuon aikakauden tietokoneissa ei vielä ollut mainittavia grafiikkaominaisuuksia, joten peli oli täysin tekstipohjainen. Al Lowe hyödynsi myöhemmin Softpornin juonikuvioita pelissään Leisure Suit Larry in the Land of the Lounge Lizards (1987), josta tulikin eräs tunnetuimpia eroottisia seikkailupelejä. Kumpaakaan näistä peleistä ei voi pitää varsinaisena pornografiana, vaikka ne aikuisille tarkoitettuja aiheita käsittelevätkin. Hieman enemmän paljasta pintaa nähtiin strip poker -peleissä, mutta suoranaista pornoa sisältävät tietokonepelit tulivat länsimaisille markkinoille vasta 1990-luvulla.

Myös varhaiset pelikonsolit saivat osansa erotiikasta. 1980-luvun taitteen markkinajohtaja Atari ei käytännössä juurikaan valvonut mitä sen konsoleilla julkaistiin, joten muun muassa pahamaineinen "raiskaussimulaattori" Custer's Revenge näki päivänvalon Atari 2600:lla. Sitä myytiin peräti 80 000 kappaletta. Konsolimarkkinat myöhemmin valloittaneiden Nintendon ja Segan kontrolli oli tiukempaa, eikä eroottisia pelejä niiden konsoleilla nähty virallisina julkaisuina.

Harrastajien keskuudessa sen sijaan levisi jo 1980-luvulla yleisesti myös bittimuotoista pornografiaa. Sisällön kopiointi oli helppoa, eikä kukaan valvonut sen julkaisua. Commodore 64:n ja Sinclair Spectrumin kaltaisten kotimikrojen grafiikkaominaisuudet riittivät jo yksinkertaisten kuvien esittämiseen. Suosittuja olivat erilaiset animaatiot ja yksinkertaiset pelit, joissa ohjattiin yhdyntää tai masturbointia peliohjaimen liikkeillä. Monien tuntema esimerkki on Landisoftin C-64:lle vuonna 1985 julkaisema Sex Games.

Digitaalisen erotiikan jäljillä

Varhaisen kotimikroerotiikan yleisyydestä huolimatta sitä on toistaiseksi tutkittu varsin vähän. Aiheeseen liittyy kuitenkin monia mielenkiintoisia kysymyksiä: mitä kanavia pitkin materiaalia levisi, miten tunnettua se oli ja millaiseksi käyttäjät



Sex Games

sen kokivat. Meitä kiinnosti ajanjakso 1980-luvulta 1990-luvun alkupuoliskolle asti, siis aika ennen CD-ROMin, internetin ja JPEG-kuvaformaatin läpimurtoa.

Näiden kysymysten selvittämiseksi laadimme kyselyn, joka julkaistiin IRC:ssä, Facebookissa sekä V2.fi-viihdesivustolla. Kyselyssä oli sekä avoimia kysymyksiä että ennalta valittuja eroottisia sisältöjä, joiden tunnettuutta halusimme selvittää. Muutaman päivän kuluessa saimme hieman alle 300 uskottavaa vastausta, joita saattoi pitää riittävänä aineistona tutkimusta varten.

Vastaajien keski-ikä oli 32,2 vuotta. Nuorin vastaaja oli 20- ja vanhin 55-vuotias, joten tutkimuksen kattamalla ajanjaksolla he olivat olleet joko lapsia tai nuoria. Kaikkiaan 94 % vastaajista ilmoitti olevansa miespuolisia, mikä sinällään noudattaa yleistä käsitystä varhaisen tietokoneharrastuksen sukupuolijakaumasta.

Kaikki vastaajat olivat tutustuneet johonkin digitaalisen erotiikan muotoon. Tämä tosin ei sinällään kerro kuin sen, että kysely valikoi vastaajiksi ihmisiä, joita aihe kiinnosti ja joille se oli ennestään tuttu. Koska kysely koski 20–30 vuoden takaisia tapahtumia, on tuloksia tulkittaessa luonnollisesti myös huomioitava, että vastaajien myöhemmät kokemukset voivat sekoittua lapsuuden ja nuoruuden aikaisiin muistikuviin. Pääosa vastauksista vaikutti kuitenkin ajanjakson tapahtumia vasten tarkasteltuna uskottavilta.

Larry, Samantha ja giffit

Voidaksemme selvittää, miten yleisesti tunnettuja tietyt eroottiset sisällöt olivat, valikoimme kyselyyn luettelon niistä. Koska mitään vakiintunutta listaa lähdekirjallisuudesta ei ollut saatavilla, perustimme valinnan ensisijaisesti omiin muistikuviimme sekä mm. Girls of C64 -sivustoon ja World of Spectrum -sivuston Sex-osioon. Vastaajat eivät nähneet luetteloa vastatessaan avoimiin kysymyksiin, mutta ennalta annetut nimikkeet ja lajityypit olivat varsin hyvin linjassa avoimissa vastauksissa mainittujen kanssa.

Monivalintaosion tunnetuin yksittäinen nimike oli Leisure Suit Larry -pelisarja, jonka tunnisti miltei jokainen kyselyymme vastannut. Tämä oli ennakoitavissa myös sarjan kaupallisen menestyksen perusteella. Eräs vastaajista koki saaneensa Larrysta peräti sukupuolivalistusta:

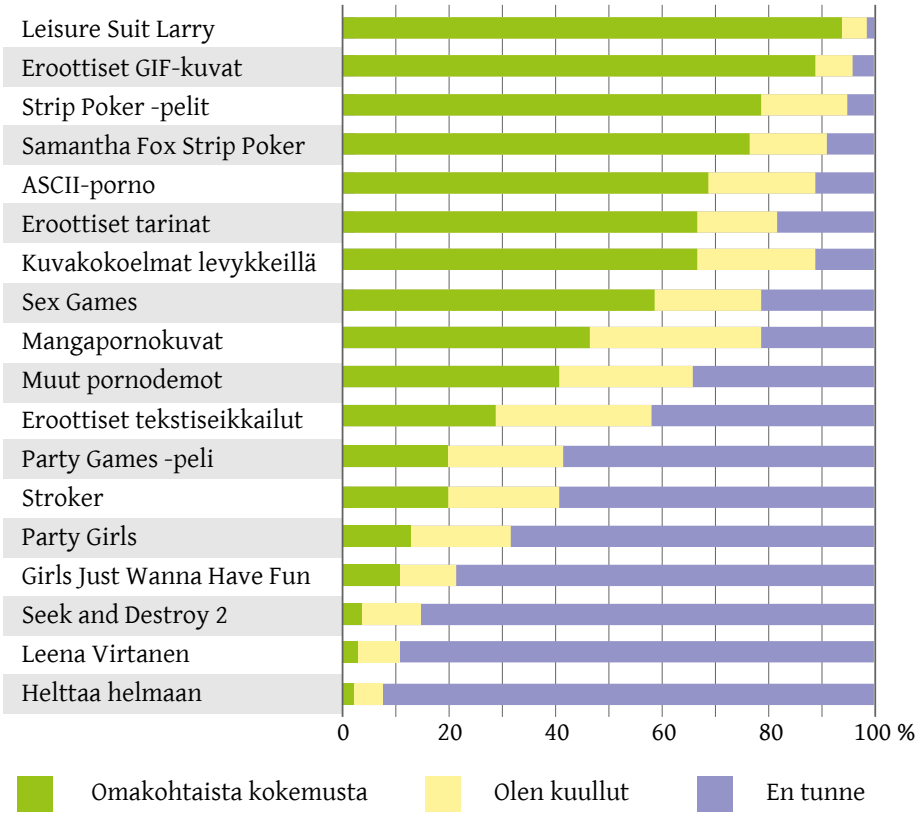
Joka tapauksessa, Larrylta tuli opittua mm. kortsuista, huorista, sukupuolitaudeista jne. ainakin nämä alkeelliset muodot. (Mies, 30)

Avoimissa kysymyksissä taas mainittiin yleisimpänä pelityyppinä erilaiset strip poker -pelit, etenkin Samantha Fox Strip Poker, jota markkinoitiin aikanaan kuuluisan laulajatähden ja Playboy-mallin yläosattomilla kuvilla. Tässä edelleen suositussa pelityypissä tarkoituksena on riisua ruudulla oleva nais- tai (harvemmin) mieshahmo voittamalla pokerissa.

Luonnollisesti paljon mainintoja saivat myös kuvat, animaatiot ja eroottiset tarinat. Tässä vaiheessa tarinat olivat etupäässä harrastajien kirjoittamia fantasianovelleja, joita vaihdeltiin BBS-järjestelmissä ja levitettiin myöhemmin internetin kautta. Skrolli-lehtikin on restauroinut näitä vanhoja tekstejä, joita oli laaja kokoelma mm. sittemmin suljetussa Zorlimin pervotekstiarkistossa.

Kuvista puhuttaessa vastaajat korostivat paljon sitä, miten niiden laatu kehittyi uusien laitesukupolvien myötä. Commodore 64:n kultakaudella suuri osa kuvista ja animaatioista oli piirrettyjä, kun taas Amiga- ja PC-tietokoneet soveltuivat digitoidun aineiston esittämiseen. Osaltaan tämä johtui myös siitä, että kuvien digitoimiseen tarvittava laitteisto yleistyi kuluttajahintaluokassa vasta 90-luvulla.

Tutkimuksen kuluessa vastaan tullutta sisältöä luonnehtii melko vahva heteronormatiivisuus. Kuvat, animaatiot, tarinat, räsypokkapelit sekä erilaiset



Kyselylomakkeessa esitellyt eroottiset sisällöt ja niiden tunnettuus.

joystickinvatkaukset kuvasivat suurimmaksi osaksi naisia tai heteroseksiä. Homoseksuaalinen vastaaja muisteleekin seuraavasti:

Lähes kaikki liikkunut porno oli heteromiehille suunnattua ja homona moisesta en saanut sitä tarkoitettua hyötyä irti. Jossain nakupokassa oli joku mieskin, mutta en muista koskaan pelanneeni sitä. Kuvia miehistä alkoi liikkua vasta purkeissa, mutta ne olivat enemmän eksotiikkaa (pitkä penis solmussa) kuin erotiikkaa tai pornoa homomiehelle. (Mies, 33)

Kokemuksia bittipornosta

Tärkeä osa tutkimustamme oli selvittää myös, miten varhainen digierotiikka levisi, millaisena varhaiset käyttäjät sen kokivat ja mitä muistoja heille siitä jäi. Aineiston avoimista vastauksista muodostuukin jonkinlainen yleiskuva siitä, miten bittipornoa saatiin ja miten siihen suhtauduttiin.

Ennen internetin yleistymistä saantikanavista tyypillisimpiä olivat ensin kaverit ja myöhemmin BBS-järjestelmät. Pornosisällöt levisivät usein samojen "solmukohtien" kautta kuin esimerkiksi pelien kopiot. Kuvakokoelmia ja pornodemoja oli tarjolla myös PD-ohjelmakirjastoissa. Joku saattaa muistaakin MikroBITTI- ja C-lehdissä ilmoitelleen Softaboxin (myöh. Avesoft), jonka mainoksissa vilisi mainintoja kuten "Animoitu Alaston demo" ja "Alastonkuvia Gonanin aikakaudelta". Kyselyvastausten perusteella näitä levyjä ei kuitenkaan moni tilannut.

Etenkään ensimmäisten kotimikrojen grafiikkaominaisuuksia ei voinut verrata painettuihin lehtiin tai VHS-elokuviin. Kuten voisi olettaa, toisten vastaajien mielestä materiaalin heikko tekninen laatu estikin varsinaisen kiihottumisen:

Tietokone-erotiikka lähinnä nauratti, enemmän sitä katseltiin huumorimielessä koska niin paljon piti mielikuvitusta käyttää niitä söherryksiä katsellessa. (Mies, 37)

Mielenkiintoinen havainto sen sijaan oli, että vähintään yhtä moni kuitenkin näki saman asian positiivisena:

90-luvulla latasin enimmäkseen suttuisia jpg-kuvia ja gif-animaatioita netistä. Ei niissä ollut enää samanlaista viehätystä kuin C-64:n animaatioissa ja peleissä. (Mies, 35)

Voidaan siis sanoa, että varhaisessa tietokone-erotiikassa yhdistyi kaksi nuoria kiinnostavaa seikkaa: kiinnostus tietotekniikkaan ja kiinnostus eroottiseen materiaaliin. Yhdistelmä teknistä viehätystä, pientä kiihottavuutta sekä aiheen kiellettyyttä saavat varhaisen digitaalierotiikan näyttäytymään joidenkin harrastajien piirissä edelleen erityisen viehättävänä ja nostalgisena.

Bittimuotoisen pornografian puolesta puhui tuohon aikaan myös se, että perheeseen hankittua kotimikroa ei useinkaan osannut käyttää kuin nuoriso. Saattoi myös olla, ettei koneen parissa puuhailu juuri muita perheenjäseniä kiinnostanutkaan. Tämä helpotti kyseenalaisen aineiston piilottamista:

Lehdet ja videot olivat parempia, mutta tietokonetavaraa oli helpompi käsitellä, koska itsellä oli tietokone omassa huoneessa (Mies, 34)

Jos käry sitten kävi, saattoi rangaistuksena olla sisällön menettäminen karullakin tavalla:

Näytimme kaverin isosiskolle ja tämän poikakaverille Sex gamesin kun se oli niin hauska/törkeä juttu nuorena teininä. Isosisko takavarikoi kasetin ja heitti sen roskiin. (Mies, 32)

Avointen kysymysten vastaukset viittaavat siihen, että digitaalinen erotiikka on nivoutunut osaksi yleistä erotiikan kulutusprosessia alusta alkaen. Vaikka tekninen laatu ei ole ollut kummoista, materiaali on koettu kiinnostavaksi ja piilottelun arvoiseksi alusta alkaen.

Ikuisuuskysymys on luonnollisesti myös se, millä tavoin varhainen kosketus pornografiaan muokkaa lasten ja nuorten käsityksiä seksuaalisuudesta. Vaikka tutkimuksemme ei varsinaisesti aiheeseen paneutunutkaan, vastauksissa tätäkin kysymystä sivuttiin:

Erotiikka muodostui mielessäni pornoa mukailevaksi. Kun sitten oikeasti aikanaan pääsin hommiin, tajusin että mielikuvani erotiikasta ja seksuaalisuudesta oli auttamattoman vähäinen ja hiukan vinoutunut. (Mies, 29)

Tikku-ukkojen lopun ajat

1990-luvun loppua kohden digitaalinen erotiikka lopulta pääosin sulautui valtavirtaan. Kuvien ja animaatioiden tekninen laatu kehittyi kehittymistään, kunnes lopulta oltiin jo nykytilanteessa: kuka tahansa voi katsoa tietokoneellaan Full HD -laatuisia eroottisia kuvia ja videoita. Henkilökohtainen tietoverkkojen käyttö poisti tarpeen kopioida sisältöjä kavereilta – katselu muuttui jaetusta kokemuksesta yksityiseksi, ja samalla saatavilla olevan materiaalin määrä kasvoi huimasti.

Eräs oleellinen taustatekijä on se, että vastaajat olivat vuosina 1980–1995 pääosin vasta lapsia tai teini-ikäisiä. Aineistossa näkyy nuorten ilmeisen kiinnostunut, mutta epävarma suhtautuminen seksiin ja seksuaalisuuteen. Erotiikan katselu on jollain tasolla tiedetty luvattomaksi, joten lapsuudenkodissa sitä on piiloteltu vanhemmilta. Porno on ollut ystäväpiirin yhteinen salaisuus, joka omalla nostalgisella tavallaan yhdistää erästä tietokoneharrastajien sukupolvea.

Artikkeli perustuu samojen kirjoittajien laajempaan tutkimusartikkeliin, joka on julkaistu Digirakkaus 2.0 -kirjassa.

Syöverin reunalla, kiikkerästi

Tapio Berschewsky

Muistatko vielä, miltä netti tuntui joitain vuosia sitten, kun fasismin lonkerot eivät vielä olleet ehtineet koskettaa juuri sinua?

Siitä ei tunnu olevan kovin pitkä aika, kun ihmiset vielä luottivat verkon suuriin palveluntarjoajiin. Kaikki oli vapaata, jos ei ilmaista, eikä mainoksia suurempia riesoja juuri kohdannut. Verkkoneutraliteetti ei toteutunut käytännössä, jos käänteli kiviä ja nuuhki ikävistä paikoista, mutta jonkinlainen yhteisymmärrys vallitsi siitä, että melkein kaikki oli riittävän hyvin.

Väitettiin, että kunhan pysyy poissa tuhmilta sivulta ja asentaa Windowsiin paremmat haittaskannerit, verkon avaama ruusuinen utopia on vaaraton. Tietoturvariskit tarkoittivat sitä, että jokin rikollisjärjestö saattaa kontrolloida etänä konettasi ja lähettää roskapostia rahan toivossa miljoonille. Nigerialaiskirjeet ja viagramainokset pelottivat ja harmittivat verkon käyttäjiä.

Nyt väestö kantaa mukanaan laivuetta NSA:n etäohjattavia kannettavia, puhelimia, ja tabletteja. Kaikki tärkeimmät palvelut ja ohjelmat ovat paljastuneet Yhdysvaltain hallituksen urkintakeinoiksi. Suojelupoliisin lausuntojen mukaan samat oman kansan vakoiluoikeudet halutaan myös Suomeen.

Pahojen rikollisjärjestöjen sijaan vaarallisin yksityisyyttäsi uhkaava vihollinen onkin valtio, kaikki valtiot, yhdessä jättikorporaatioiden kanssa. Tervetuloa dystopiaan, jossa jokainen hakutermisi voi tuomita sinut myöhemmin.

Vähennä altistumistasi

Juuri ennen kolumnin kirjoittamista julkaistiin uutinen siitä, kuinka Saksan tiedustelupalvelun mukaan turvatoimenpiteenä tunnettu trusted platform management -piiri mahdollistaa Windows 8 -koneen erittäin laajan etähallinnan NSA:lle. Applen laitteissa ei tpm:ää ole, eikä Linux tue ominaisuutta.

On helppo hymyillä, kun viimeiset seitsemän vuotta on käyttänyt kaikilla omilla tietokoneilla vain Linux-jakeluja. Kun verkko yhä tuntuu turvalliselta ja ohjelmat luotettavilta. Omaan linnakkeeseen tosin ei pitäisi tuudittautua, kun samaan aikaan huonompia päätöksiä tekeviä kanssaihmisiä nussitaan takalistoon kuumalla sapelilla.

Vaan mitä voi tehdä, jos on sapelin vastaanottaja? Ainakin vaihtaa käyttöjärjestelmää. Microsoft ei ole ystäväsi, vaan myy sinut sentistä ensimmäiselle painostajalle. Epäilisin myös Applea. Vain avoimeen koodiin voi luottaa. Ei näin pääse kuin alkuun, mutta ainakin lujalle pohjalle voi rakentaa kestävän talon.

> *Internet on nykyään perseestä.*

Seuraavaksi hankkiudutaan eroon PRISM -puhelimista. Ei Androidia, ei iOS:ää, ei Windows Phonea. Jos älyluurin tarvitsee, Jolla on ainoa vaihtoehto. Tai ehkä Jolla ja Ubuntu Phone. Aina voi myös vaihtaa mummoluuriin ja kantaa mukana jotain luotettavamman oloista tietolaitetta verkon käyttöön.

Niin, hankkiutukaa eroon myös Chromesta, Chromiumista, ja Googlesta ylipäätään. Hakekaa Duckduckgo:lla ja asentakaa parempi selain. Jos Chromiumin hyvistä puolista ei halua eroon, esimerkiksi SRWare Iron pohjautuu samaan projektiin, muttei vuoda salaisuuksiasi hyväksikäyttäjille. Myös Tor Browser Bundle on erittäin helppo ottaa käyttöön. Näitäkin tosin pitää osata käyttää varoen.

Vain vakoilijan silmille

Niinhän se vain on: internet on nykyään perseestä. Kaikki parhaat palvelut, joita haluaisi käyttää, ovat tietoturvan puolesta rikki. Silti niitä tulee jonkin verran käytettyä. Minullakin on fecesbook-tunnari, ja olen ainakin 20 kertaa sanonut sinne jonkin henkilökohtaisen mielipiteen jostakin. Löytyy myös gmail, plussa, twitter, ja mitä muita näitä nyt on. En juuri käytä, mutta olen NSA:n kartalla silti.

Poikkeuksena työasiat, joissa Google on ikävä kyllä kovalla käytöllä. Drive ja Docs ovat niin käteviä palveluja toimittajalle, että niitä on vaikea välttää. Skrollikin tehdään Driveä käyttäen. Kaikki artikkelimme ovat siellä viimeistään kielenhuoltoa varten. Monet niistä on jopa kirjoitettu suoraan Googlen tekstinkäsittelyä käyttäen. Avointa ja vakoiluvapaata versiota odotellessa — pelkkä kirjoitusalusta.fi ei riitä.

Aina välillä tekisi mieli pistää pois päältä kaikki sähkölaitteet, riisua vaatteet ja juosta metsään kiipeilemään puissa. Sitä kutsutaan kesäksi. Näin syksyisin taas suorittimen lämpö kiinnostaa liikaakin. Pahassa maailmassa elämisestä voi repiä lohtuakin, jos osaa ajatella positiivisesti. Ainakin NSA lukee Skrollia, jos muut eivät.

Merkittäviä roolipelejä

Roguen perintö on jatkunut jo kolmanneksen vuosisadasta. Merkkigraafinen sankarointi ei kuole pois.

Teksti: Kalle Viiri
Kuvat: Mitol Berschewsky, Kalle Viiri

Roguelike on peligenre, joka muistuttaa toimintaroolipeliä tietyillä erikoispiirteillä. Pelaaja ohjastaa hahmoaan maanalaisissa luolastoissa ja käytäväverkostoissa. Tavoitteena on selvitä hengissä kohtaamisista maanalaisten hirviöiden kanssa ja kerätä talteen syvälle maan uumeniin kätketyt aarteet.

Roguelikelle tyypillistä on, että hahmon kuolema on lopullinen, ja pelialueet hirviöineen ja aarteineen ovat satunnaisesti luotuja. Juoni on yleensä läsnä vain nimellisesti. Luolaston pohjalla makoilee hirviöiden vartioima kallisarvoinen amuletti tai muu taikaesine, joka pitää hakea maan pinnalle.

Genreen yhdistetään usein merkkigrafiikka, vaikka vaihtoehtoiset grafiikat ovat yleistyneet jatkuvasti.

Varhaishistoriaa

Nimitys juontaa englannista: *Rogue like* eli "Roguen kaltainen". Vuonna 1980 julkaistu Rogue popularisoikin genren sisältämällä kaikki sen pääpiirteet. Mukana olivat satunnaisgeneroitu luolasto, kevyt juoni, jatkuvasti ruokaa tarvitseva sankari ja anteeksiantamaton kalman käsi, joka viedessään hahmon pitää sen otteessaan ikuisesti.

Pelin suosio oli niin valtava, että Rogue lisättiin vakio-osaksi suosittua BSD 4.2 -käyttöjärjestelmää. Roguelike-ilmiön perusta oli luotu.

Roguelike-genren muotoutuminen tapahtui pääasiassa 1980-luvulla Roguen inspiroidessa uusia pelejä. Ensimmäisenä apajille ehtivät Moria ja Hack, jotka samalla polkaisivat käyntiin kaksi rogueliken päähaaraa: "bandit" ja "hackit".

"Bandien" esikuvassa Moriassa luolasto on loputon, sillä peli generoi uusia alueita aina pelaajan siirtyessä uudelle kerrokselle. Pelaaja voi myös varustella hahmoaan rauhassa maan pinnalla olevassa kaupungissa.

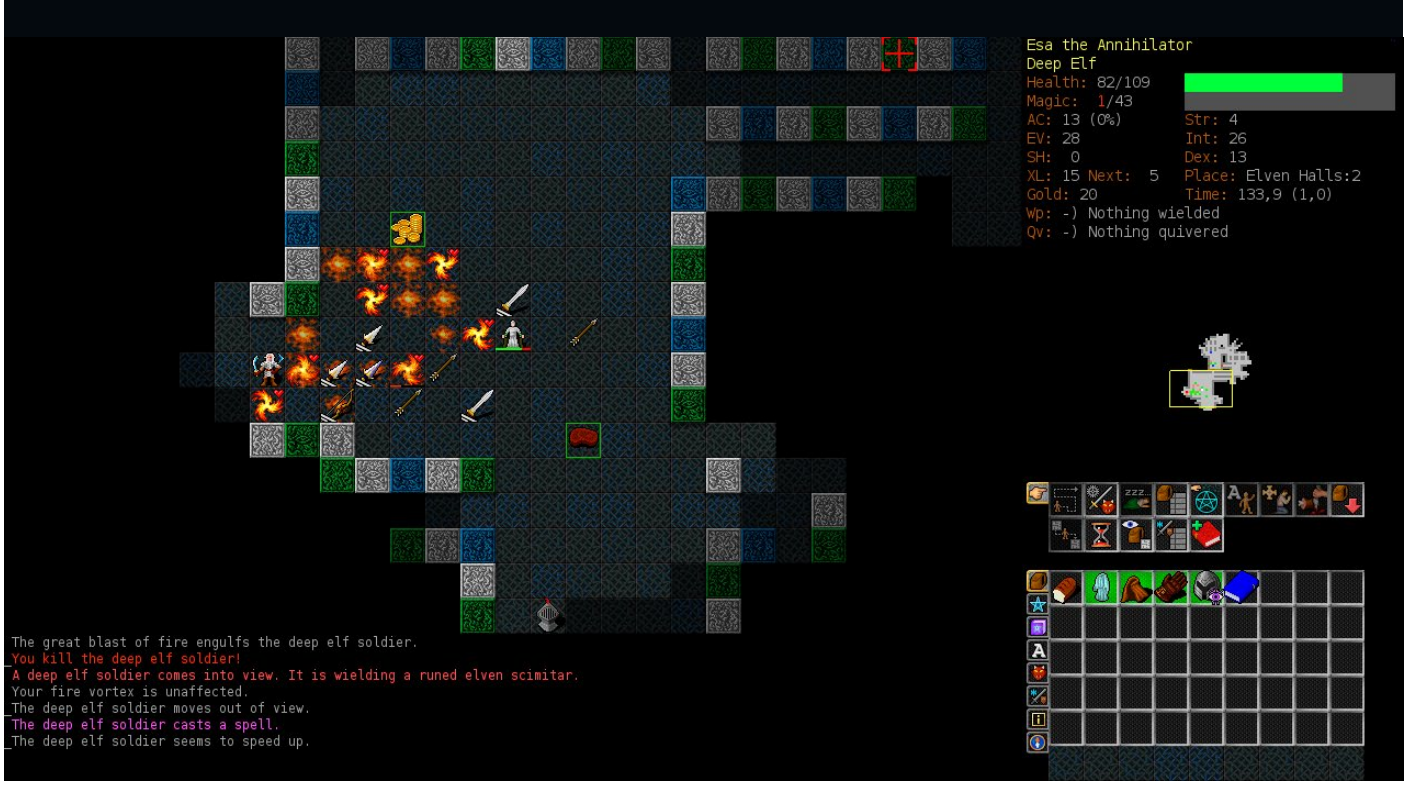
Maan alta löytyy hirviöitä ja niiden vartioimia aarteita, jotka auttavat hahmon valmistamisessa taisteluun luolaston pohjalla lymyilevää Balrogia vastaan. Pelin pääpaino on hahmon kehittämisessä. Moriasta jalostettiin kehitetty versio Angband, joka itsessään on kloonattu sadoiksi eri variaatioiksi.

Hack on Roguelle uskollisempi klooni, joka lisää peliin erilaisia hirviöitä, alueita ja ominaisuuksia. Se säilyttää kuitenkin pelityylin samanlaisena: etene luolastossa, kerää aarteita, tapa hirviöitä, koeta selvitä. Turvasatamia on harvassa, ja nälkäkuoleman vaara rankaisee etenemistä arkailevaa pelaajaa.

Peli on pullollaan mytologian ja popkulttuurin inspiroimia ansoja, jotka uusi pelaaja oppii usein kantapään kautta. Hack muistetaan nykyään parhaiten Usenetin välityksellä kehitetyn NetHackin esiasteena.

Vanhemmista roguelikeista löytyy myös muutama erikoisempi nimi. Larn

Nethack on ehkä tunnetuin roguelike, jopa tunnetumpi kuin Rogue itse. Tässä koboldi ottaa nenään sankarilta.



Dungeon Crawl Stone Soup on tämän hetken parhaita roguelikeja. Mukavan käyttöliittymän ja lempeämmän pelisuunnittelun ansiosta tämä uppoaa paremmin moderniin pelaajaan kuin klassisemmat teokset. Pelistä on myös merkkigraafinen versio.

muistuttaa ensi vilkaisulla hieman Moriaa ja Angbandia, mutta mahdollisuutta kehittää hahmoaan rauhassa ei ole. Pelillä on tiukka aikaraja jonka puitteissa jalokivi "Larnin silmä" on saatava ulos luolastosta.

Omega puolestaan tarjosi ensimmäisenä roguelikena pelaajalle laajan maailman ja avoimen pelityylin. Kömpelön käyttöliittymän, bugisten julkaisujen ja erikoisten pelimekaniikkojen ansiosta Omega ei kuitenkaan saanut aikaan varsinaista läpimurtoa.

Yhdistävät tekijät

Klassisten roguelikejen välillä on paljon eroja, mutta yhdistävät tekijät ovat vielä korostuneemmat. Eniten pelikokemukseen vaikuttaa lopullinen kuolema, pelottava *permadeath*.

Kuoleman lopullisuus tarkoittaa, että jos pelaajan hahmo kuolee, pelaaja joutuu aloittamaan pelin alusta. Se tekee peleistä paljon vaikeampia. Klassista NetHackia pelataan usein vuosia ennen kuin luolaston pohja alkaa häämöttää.

Vaikka kuoleman lopullisuus tuntuu joskus liian rankalta, se lisää peliin paljon syvyyttä. Pelaaja ei voi tukeutua siihen, että huonosti käydessä voi palata pari minuuttia vanhan tallennukseen. Pitkälle kehittyneen hahmon joutuessa tiukkaan paikkaan syke nousee ja jännitys on käsinkosketeltavaa.

Useimmat rogueliket ovat vähäisen juonensa ja satunnaisesti luotujen ympäristöjensä takia miellyttäviä aloittaa alusta vanhan hahmon kuoltua. Pelaajan ei tarvitse kahlata samoja taisteluja ja käytäviä, vaan ympäristö on aina uusi.

Rogueliket mielletään hankaliksi peleiksi, joista ei selviä ilman strategiaopasta tai pitkää yrityksen ja erehdyksen kautta tapahtuvaa oppimista. On totta, että monet roguelikeistä jakavat äkillisiä ja arvaamattomia kuolemia avokätisesti.

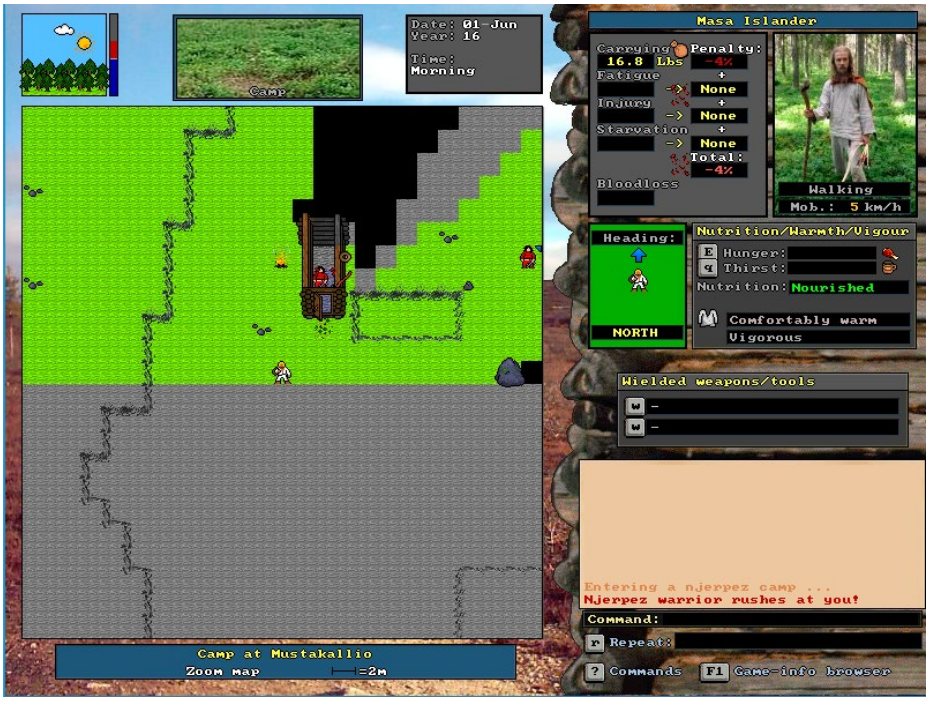
Erityisesti NetHackissa erilaisia odottamattomia vaaratilanteita on runsaasti myös heti pelin alussa. Jopa niinkin arkinen asia kuin suihkulähteestä hörpyn ottaminen voi koitua hahmolle kohtalokkaaksi.

Monille nykyaikaisempiin peleihin tottuneille on myös vastenmielistä omaksua perinteinen merkkigrafiikka ja oppia pelin lukuisat oleelliset näppäinkomennot.

Kyllä se kehittyy

Roguelikejä on kuitenkin tehty 1980-luvun jälkeenkin, ja etenkin 2000-luvun puolella tehdyissä peleissä näkyvät selvästi myös kehityksen merkit. Modernisointia tapahtuu monella tavalla.

Käyttöliittymiä kehitetään, aloittelijaystävällisyyttä lisätään, vanhoista perinteikkäistä luolaympäristöistä astutaan seikkailemaan avoimeen maailmaan, tai scifihenkiseen tulevaisuuteen



Brutaalin realistinen Unreal World tarjoilee tylyjä kokemuksia muinaispohjolan luonnossa. Susista ja muista vaaroista selvinnyt kulkija on tunkeutumassa leiriin.

sivuhahmoineen ja -tehtävineen. Mitä roguelikeille oikein tapahtui 1980-luvun jälkeen?

Omegan aikanaan luotsaama avoin maailma sivutehtävineen sai uuden mahdollisuuden 1990-luvulla. Suosituksi tulleet Angband-variantit ZAngband ja ToME sekä NetHackin ja Omegan inspiroima ADOM astuivat kuvioihin.

Vaikka keskeisin sisältö tapahtuu edelleenkin luolissa, maan päältäkin löytyy taisteluita, kaupunkeja ja sivutehtäviä. Eteneminen muuttuu vähemmän lineaariseksi, kun luolastossa syvemmälle siirtymisen sijasta pelaaja voikin palata pinnalle puuhastelemaan.

ADOMin myötä myös juonivetoisen rogueliken suosio on kasvanut. Juonivetoisuus on rogueliken kehittäjälle haaste. Juoni ei saa häiritä pelaajaa, vaikka sen joutuisi kokemaan kymmeniä kertoja hahmojen kuollessa.

Yleinen tapa tuoreuden säilyttämiseksi on, että osa pelin sisällöstä pysyy pelikerrasta toiseen samana, ja osa taas luodaan satunnaisesti jokaista pelikertaa varten. Näin tekevät ADOMin lisäksi esimerkiksi suomalaistekoinen IVAN ja erikoisempi, mechataisteluun keskittyvä Gearhead.

Perinteisemmissäkin roguelikeissä on tapahtunut uudistuksia. Esimerkiksi Dungeon Crawl on klassinen luolastoseikkailu ilman juonta, kaupunkeja tai kilttejä sivuhahmoja. Samalla se pyrkii poistamaan pelistä roguelikeen usein yhdistettyjä ikäviä piirteitä kuten mekaanisen toiston, äkkikuolemat, ja vain kantapään kautta opittavat ansat. Osaltaan Dungeon Crawlia voikin pitää vastalauseena sekä pitkällistä hahmonkehitystä suosivalle Angbandille että uusia pelaajia jatkuvasti kampittavalle NetHackille.

IVAN mallintaa monia asioita tarkemmin kuin on roguelikeissä totuttu näkemään. Pelin suurin kompastuskivi lienee se, että peli vaikeutuu sitä mukaa kun pelaajan varustelu paranee – usein kannattaa jättää vaihtamatta parempaan aseeseen.

Myös klassinen räiskintäpeli taipuu roguelikeksi. DoomRL:ssä väkivaltafantasiaa pääsee toteuttamaan vähän toisesta näkökulmasta kuin vatsaan pultatun BFG9000:n takaa.

Kun genre ei riitä

Roguelike on nostanut viime aikoina päätään indie-pelien maailmassa inspiraation lähteenä. Perinteisestä luolastoseikkailusta voidaan poiketa runsaastikin, mutta klassisista roguelikeista haetaan ideoita niin ulkoasuun kuin pelimekaniikkoihinkin.

Yksi ehdottomasti tunnetuimpia genrestä ulos rönsyäviä roguelikejä on Dwarf Fortress. Pelin seikkailutila muistuttaa paljon normaalia roguelikea, mutta pelissä on myös linnoitustila, joka muistuttaa enemmän kaupunginrakennuspelin ja rogueliken risteytystä.

Dwarf Fortress on osaltaan toiminut inspiraation lähteenä ja pioneerina realismia ja selviytymistä painottavissa roguelikeissa. Muita vastaavia ovat esimerkiksi Cataclysm: Dark Days Ahead ja suomalaista tekoa oleva, jo 21-vuotias UnReal World.

Roguelike on saanut osakseen kaupallista suosiota Japanissa, jossa luolastojen aarteita pengotaan useammankin kaupallisen pelisarjan voimin. Ilmiön pioneerina siellä pidetään Mystery Dungeon -sarjaa, jonka lisäksi markkinoilla on muun muassa Final Fantasy -pelien sivuosana toimiva Chocobo-sarja.

Länsimaissa roguelike on jäänyt pääosin kaupallisen valtavirtapelikehityksen ulkopuolelle. Genren ominaisuuksia ja nimeä lainataan ahkerasti indie-puolella, jossa leikitellään mieluusti satunnaisten maailmojen ja lopullisen kuoleman kanssa. Esimerkiksi suosittu avaruusseikkailupeli Faster Than Light tunnetaan roguelike-elementeistään.

Vaikka genren valtavirtasuosio on vielä vähäistä, kehitysskene on aktiivinen. Uusia roguelikeja saatetaan maailmaan esimerkiksi "Seven Day Roguelike" -kilpailuissa, joissa nimensä mukaisesti ohjelmoidaan roguelike seitsemässä päivässä. Merkkigrafiikkaa tai ei, Roguen perintö on tullut jäädäkseen.

IRC on paras

25 vuotta tekstiä

Juttelualustoja netissä tulee ja menee, mutta irkki on kestotavaraa.

Teksti: Ronja Koistinen Kuva: Mitol Berschewsky

IRC eli Internet Relay Chat, kavereiden kesken irkki, on oululainen keksintö 80-luvulta. Sillä on ainakin puoli miljoonaa käyttäjää ympäri maailmaa. Määrä on pikkuruinen vaikkapa Facebookiin verrattuna, mutta irkki on tärkeä erityisesti Suomessa ja tietyissä piireissä muuallakin maailmassa.

Missään muussa chatissa kuin irkissä ei ole yhtä selkeää ja yksinkertaista jutella joko ryhmille tai yksittäisille käyttäjille. Kanavia on helppoa perustaa, ja niiden toiminta on idioottivarmaa. Kanavat voivat elää yhden illan tai vuosikymmeniä. Niiden pohjana voi olla kaveripiiri, ohjelmistoprojekti tai vaikka paikkakunta. Esimerkiksi Skrollilla on useampi irc-kanava, kuten ircnetissä sijaitseva #skrolli, jossa lehteä kehitetään.

Ei mitään ylimääräistä

Irkissä parasta on selkeys ja yksinkertaisuus. Tekniikka on suoraviivaista ja standardi avoin. Irkki on ihan oikea, oma protokollansa ja palvelunsa.

Irc-verkkoja ylläpidetään usein hajautetusti monien toimijoiden yhteistyöllä. Eri verkoissa on kuitenkin erilaiset käytännöt niin palvelimien ylläpidollisten velvoitteiden kuin kanavien sisäisen ylläpidonkin suhteen.

Asiakasohjelmia on monia erilaisia, joista valita mieluisensa. Suomessa internet-palveluntarjoajan oma IRCnetissä oleva palvelin on osa peruspalvelua. Verkkoja on monia muitakin, niin kansainvälisiä kuin paikallisia. Pienimmät irkkiyhteisöt ovat yksittäisten omien palvelinten ympärille kasautuneita tuttavapiirejä.

Irkissä on myös mahdollista olla juuri niin anonyymi, kuin hyvältä tuntuu. Käyttää voi nimimerkkiä tai koko nimeä, ja jotkut verkot tarjoavat jopa palveluita IP-osoitteen piilottamiseen muilta irkkaajilta.

Facebook ja Google+ nyrpistelevät kovasti nenäänsä tekaistuille nimille, ja monelta suunnalta tulee muutenkin painostusta oman nimen käyttöön. Irkissä voi kuitenkin periaatteessa toimia niin kuin tykkää, vaikka joillakin kanavilla oikean nimen näyttäminen lasketaan osaksi hyviä käytöstapoja.

Avointa ja julkista

Irkin tietoturvaominaisuudet vaihtelevat rankastikin eri verkkojen ja palvelinten välillä. Käyttäjien IP-osoitteet näkyvät useissa verkoissa avoimesti, eikä nimimerkkien autentikointikaan ole käytössä kaikkialla.

Monet verkot tukevat yhteyden muodostamista SSL-salattuna. Salauksesta on kuitenkin hyötyä vain, jos jokainen keskusteluun osallistuva henkilö käyttää salausta. Jos yksikin linkki näppäimistön ja irc-palvelimen välillä on yhdeltäkin käyttäjältä salaamatta, koko keskustelu on luettavissa selkokielisenä verkkoliikenteestä.

Käyttäjä ei myöskään voi helposti varmistaa, onko irc-palvelimien välinen liikenne salattua vai ei. Parasta suojaa saa siis hyvin konfiguroiduilta ssl-suojatuilta pienen piirin palvelimilta, jossa kaikki käyttävät salausta.

Lopulta irkkiä kannattaa kohdella julkisena keskusteluna. Monet käyttäjät tallentavat kaiken keskustelun omille koneilleen. Ei ole siis mitään takeita siitä, mihin irkkiin sanotut asiat saattavat päätyä.

Viranomaisten tiedetään tarkastaneen irkkaajien käyttäjätietoja selvittäessään rikosten taustoja.

Korvaajaa odotellessa

Vuosien varrella irkille on ehdotettu ja kehitetty useita korvikkeita. Paremmasta päästä on SILC, joka ei kuitenkaan ole saavuttanut suosiota. Se hyödyntää vahvaa kryptografiaa ja avainten vaihtoon perustuvaa käyttäjien keskinäistä luottamusverkostoa, jollaisen kunnollinen ylläpito vaatii vaivaa ja pedanttisuutta.

Silccaajien käyttäjänimet eivät ole uniikkeja. Kahden samannimisen käyttäjän erottamiseksi täytyy erikseen puljata salausavainten kanssa. Irkissäkin tosin käyttäjänimien kaappaaminen on arkista, eikä pelkkään nimeen kannata luottaa.

Irkissä viehättää sen elegantti yksinkertaisuus. Facebookin kaltaisissa moderneissa suurissa sosiaalisissa medioissa on äkkiä pyörällä päästään monimutkaisista yksityisyysasetuksista, kaveripiireistä ja ryhmistä. Välistä on vaikea hahmottaa, mihin asti mikäkin postaus näkyy. Kaikessa verkkoviestinnässä on tietysti mahdollista olla ylimääräisiä korvia, mutta irkissä on sentään selvää, keille kirjoituksensa osoittaa.

Mikään firma ei ole myöskään koko ajan tunkemassa väliin mainoksineen, eikä kukaan järjestele ja suodata sisältöä uuteen järjestykseen jonkin algoritmin perusteella. Irkki avautuu irkkaajan silmien eteen sellaisenaan, eikä siinä tapahdu mitään muuta kuin ihmiset itse.

Päästä sisäinen koodarisi irti!

Raspberry Pi Starter Kit

Yksinkertainen yhden piirilevyn tietokone. Sen on kehittänyt Raspberry Pi Foundation, jotta useammat saisivat mahdollisuuden oppia tietokoneen tekniikkaa ja ohjelmointia ilman suurempia sijoituksia.

Tässä paketissa on kaikki mitä tarvitset Raspberryyn tutustumista varten:

- Raspberry Pi -piirilevy
- SDHC- muistikortti
- Virtalähde
- Kotelo piirilevylle
- Jäähdytyssiilit

79 €

Jimm's Custom Perfect Gentleman

Kun saavut laneille, tahdotko saada täydellisen herrasmiehen vastaanoton?

Perfect Gentleman on suunniteltu kompaktiin BitFenix Prodigy koteloon, joten sen ulkoiset mitat taistelevat jopa Marilyn Monroen mittojen kanssa. Erittäin pieni, ellei täydellinen!! Prodigy koteloon on silti onnistuttu ahtamaan tehoja jopa Espoon tarpeisiin!

Tekniset tiedot:

- Intel i5-4670K 3.4GHz, 6MB -prosessori
- Gigabyte GA-Z87N-WIFI mITX -emolevy
- NVIDIA GeForce GTX 760 2GB GDDR5 -näytönohjain
- 8GB 1600MHz DDR3 CL9 -keskusmuisti
- 120GB SSD-kiintolevy + 1TB SATAIII -kiintolevy
- BitFenix Prodigy Midnight Black, Mini-ITX -kotelo
- Cooler Master Seidon 120M vesijäähdytysjärjestelmä
- Windows 7 Home Premium 64-bit (Suomi)

1389 €

JIMM'S
www.jimms.fi

Asiakaspalvelu
+358 29 70 70700
asiakaspalvelu@jimms.fi

Jimm'siltä löytyy nettitilauksiin aina toimituskuluton vaihtoehto!

Noutopalvelumyymälä Turku
Lukkosepänkatu 7
20320 Turku

Hinnat €/kpl, sis. alv24%